Received: February 19, 2023. Revised: April 23, 2023.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Multiple Resource Attributes and Conditional Logic Assisted Task Scheduling in Cloud Computing

Karnam Sreenu[1]* **Sreelatha Malempati[2]**

[1]*Department of Computer Science and Engineering, ANU College of Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India*
[2]*Department of Computer Science and Engineering, RVR & JC College of Engineering, Guntur, Andhra Pradesh, India*
* Corresponding author's Email: karnam.sreenu@gmail.com

Abstract: From the past few decades, the cloud computing has been arisen as an extensively used platform to provide storage, compute and analytics services to organizations and end users on the basis of pay-as-you-use. This enabled the organizations as well as individuals to access the larger set of resources without establishing a costly and high performance computing platform. However, the major issue in cloud computing is the task scheduling which is declared as NP-hard problem and most of the researchers applied meta-heuristic algorithms to solve it. However, they experienced a slower convergence speed which has a direct impact on the efficiency of cloud computing environment. To achieve a faster convergence along with efficient quality of Service, this paper proposes a simple and effective task scheduling mechanism based on multiple resource attributes and conditional logic. This method considers totally four resource attributes such as Resource reaction time resource location, resource availability and resource reliability rate. Based on these four attributes, the proposed method constructs an index called as task scheduling index (TSI) and assigns tasks for resources based on their TSI value. The TSI is constructed through the proposed conditional logic and high priority is given for resources those have higher TSI. For experimental validation, we used two benchmark datasets such as GOCJ and synthetic dataset. Three performance metrics namely Makespan, Throughput and convergence speed are measured and compared with state-of-the-art methods like PSO, GA and GWO. On an average, the reduced Makespan of proposed method is 48.22% and 46.87% for GOCJ and Synthetic datasets respectively.

Keywords: Cloud computing, Task scheduling, Convergence, Meta-heuristic, Conditional logic, Thought and makespan.

1. Introduction

Recently, the cloud computing has become a popular and attractive platform for different organizations those provides the resources for the execution of different large scale applications [1, 2]. Further, the cloud computing put one step forward and initiated to provide the services for users based on their requirements. Such kind of provision is called as on-demand resource sharing and it is an essential requirement for the Internet based applications. Depending on the service required, the cloud services are categorized into four categories; they are "Platform as a service (PaaS)" [3], "Infrastructure as a service (IaaS)" [4], "Expert as a service (EaaS)" [5], and "Software as a service (SaaS)" [6]. Cloud computing enables the parallel and distributed computing by proving both hardware and software resources on a shared basis like as "Pay-as-you-go" [7]. To uses the resources, the users need an internet connection and they have to purchase any service or platform. Due to these many advantages and flexibilities, the cloud computing has become an emerging paradigm.

Cloud can be deployed in three modes; they are private, public and hybrid cloud platforms [8, 9]. For the organizations those can maintain own private cloud, the cloud computing provides an opportunity to use the cloud resources in hours when user's demand is at the peak. On the other hand, the public

clouds are freely available and easily accessible [10]. Next, the community clouds are defined as the ones maintained by the organizations while the hybrid clouds use the resources of both private and public clouds. For any kind of mode, the cloud service provides needs to efficiently manage the cloud resources such that there would be an improvement in the performance in different applications. For any kind of applications, the performance improvisation is achieved with larger number of resources, however, it increases the resources usage cost.

One of the important and tough task in cloud computing is task scheduling which has great impact on the cloud in terms of "Quality of service (QoS)" provision [11]. The task scheduling ensures a balanced execution between the user requirements and resources utilization. Every task needs different requirements of memory, computational time and response time as there exists different types of users. IaaS is one such kind of service through which the cloud service providers can provide server or cloud computer for storing and processing the data in the cloud. Users are eligible to access the free services of by running their applications over a rental server of a cloud computer. Since every user pay to use the cloud resources, they expect an excellent QoS. An improper task scheduling dissatisfies the users and May results in several serious constraints. Hence, the task scheduling becomes an important task in clod computing.

As there exists different kinds of users those seeks the cloud resources, the task scheduling is considered as an NP-Hard problem. To solve this problem, recently many authors tried to deploy meta-heuristic algorithms by formulating the task scheduling as a main objective function. However, the Meta-heuristic algorithms constitute several problems like

1. Parameter tuning: In meta-heuristic algorithms, there exist several dependent and independent parameters through which the objective function is optimized. This is an iterative process and even for one optimal solution, it consumes larger time which indirectly results in convergence problem.

2. Additional resources requirement: As there exists typical mathematical operations for each Meta-heuristic algorithm, the CSPs need additional resources to compute them.

To sort out these problems, this paper introduces a new and simple task scheduling mechanism based on multiple objectives and conditional logic. For every user request, our method calculate one index called as task scheduling index (TSI) based on the four resource attributes such as reliability rate, availability, reaction time and location. The major contributions of this paper are outlined as follows:

1. Formulate mathematical modelling for the calculation of reliability rate, availability, reaction time and location.
2. Propose a new task scheduling index (TSI) that considers the multiple resources attributes as reference parameters to decide the capability of resource to handle the task.
3. Propose a simple conditional logic assisted rules to formulate the TSI for each resource provider.

Remainder of the article is prepared as follows; 2nd section explores the details of literate survey. 3rd Section explores the particulars of proposed methodology. 4th Section provides the details of experimental validation and section V provides the Conclusion details.

2. Literature survey

This section outlines different task scheduling methods and categorizes them into two broad categories. They are task scheduling through QoS parameters and task scheduling through Meta-heuristics. Under second category, we again sub-categorize them into various categories based on the algorithm deployed.

2.1 Task scheduling based on QoS

Under this category, the different QoS parameters are taken into picture to assign a task to a particular device [12]. Wu et al. [13] considered the QoS parameter called as "Most time to complete the task (MTCT)" and proposed a QoS based task scheduling algorithm in cloud computing. However, they didn't guarantee about the reliability and location errors. Albodour et al. [14] considered the business cloud actions relative to QoS and rescheduling capabilities to schedule the tasks. They proposed a theory of "Business Gird QoS (BGQoS)" and analyzed the conduct of several components and actions within the BGQoS. The rescheduling capabilities increases delay thereby further users suffer with waiting time. Ali et al. [15] divided each task into five categories based on task latency, task size, task type. Then they scheduled a grouped task scheduling based on QoS metric in cloud computing. Sabu M. Thampi et al. [16] developed a task scheduling algorithm based on the awareness of QoS cost. They considered the virtual resources related to QoS which are available from real time unified resource layer. They deployed a "virtual machine manager (VMM)" which updates about the state of art information about resources to take an appropriate decision. However, they have considered only limited factors to assess the capacity

of could resources. Unlike these methods, the proposed approach totally considered four factors which have different broad contexts in determining the capability of resource.

Belal Ali AL-Maytami et al. [17] proposed a task scheduling algorithm based on "Directed acyclic graph (DAG)". They computed the "Prediction of tasks computation time algorithm (PTCT)" to identify the prominent cloud. Additionally, they proposed an algorithm to improve makespan and reduce the complexity through "Principal component analysis (PCA)" and diminished the "Expected time to compute (ETC)" matrix. Reliability rate is not used to assess the resource capacity due to which the task execution takes more time for complete. Hadeer Mahoumd et al. [18] performed task scheduling by considering three QoS attributes such as computation cost, earliest finish time and total length of task using decision tree algorithm. They executed their method in three phases; they are priority task, resource matrix, and resource allocation. The first phase assigns a rank for every task and second phase is employed for the collection of task features while the last phase allocates the virtual machine based on decision tree. Even though three different factors are used to assess the resource's task handling capacity, the availability and location errors are not determined. Such process results in less throughput and less make span.

2.2 Task scheduling

Under this category, the task scheduling is done through meta- **Meta-heuristic based** heuristic algorithms. For all these methods, the entire methodology is executed in two phases; they are fitness function formulation and optimization in an iterative fashion. Several types of algorithms are used here to optimize the fitness function through different strategies.

2.2.1. Ant Colony optimization (ACO)

ACO utilizes the positive criticism and imitates the behavior of ant colonies for food scanning and interaction based on the pheromone laid on the voyage way. Qiang Guo [19] formulated the fitness function based on makespan and cost of the tasks. Next, they applied ACO to optimize the fitness function and to ensure a load balancing between different tasks. Wei X [20] formulated the fitness function called as task scheduling satisfaction function based on three attributes such as task completion cost, degree of resource load balance and shortest waiting time. Further, they improved the ACO by involving a new coefficient called as virtual machine load weight coefficient and used to update the process of local pheromone. Gang Li et al. [21] focused on the load imbalance problem in "System Wide Information Management (SWIM)" task scheduling. They referred load standard deviation function and hardware performance quality index resources to update the pheromone at ACO in task scheduling. Hongji Liu [22] focused on the improvisation of Convergence speed and proposed a "Polymorphic ACO (PACO)" for task scheduling in cloud computing. PACO optimizes the fitness function formulated based on balanced load rate, lower cost and shorter execution time. Elsayed Elsedimy et al. [23] also aimed at convergence speed and proposed an improved ACO called as multitask objective task scheduling ACO (MOTS-ACO). They promoted the diversity of the Pareto set and incorporated the adaptive distribution probability.

Even though ACO is determined as a best solution in solving the discrete problems, it has several inevitable disadvantages; they are less convergence speed and less accuracy when dealing with data with huge dimensions. Since cloud computing is linked with huge data, the ACO introduces an excess delay in task execution due to slow convergence rate.

2.2.2. Particle swarm optimization (PSO)

PSO algorithm is one of mostly employed Meta-heuristic algorithm and it works based on the behavior of birds. N. Dordaie [24] proposed the Hybrid PSO by incorporating Hill climbing algorithm for task scheduling in cloud environments. Xingwang Huang et al. [25] proposed a task scheduling model to minimize the make span. They designed task scheduling through 5 discrete variants of PSO which have differences in their inertia weight updating strategies. Among the five one is linear and remaining four are discrete versions and used for the assignment of virtual machine for each task. Heba M. Eldesokey et al. [26] proposed a hybrid swarm optimization by combining PSO with "Salp swarm optimization (SSO)" to resolve the task scheduling problem in cloud environments. They formulated the fitness function based on two attributes such as computation cost and execution time. They also employed a "Multilayer regression (MLR)" to detect the overloaded CMs such that the task burden can be distributed in a uniform way.

Zhou Wu et al. [27] modeled the fitness function based on Execution time and applied modified PSO [28] for task scheduling in cloud environments. A Copula function is defined to alleviate the

interconnection of random number random parameters and introduced a local attractor to make the fitness function get escaped from trapping into the local minimum. Said Nabi et al. [29] contributed an Adaptive PSO assisted task scheduling by introducing an novel strategy for updating the inertia weights called as "Linearly descending and adaptive inertia weight (LDAIW)" for task scheduling in cloud environments. M. Sudheer et al. [30] combined PSO with cuckoo search to perform ask scheduling based on the priority of VM and Task in cloud environments.

Even though PSO has gained great importance in different fields at solving the optimization problems, it makes the system to suffer from local optimum problem for high dimensional space. Further, it also has low convergence rate in the iterative process.

2.2.3. Other Methods

Since there exists so many meta-heuristic algorithms, the researchers used different algorithms for implementing the task scheduling in cloud environments. Some of the other algorithms include "Genetic algorithm (GA), artificial bee colony (ABC) and grey wolf optimizer (GWO)" etc. Deafallah Alsadie [31] employed GWO for the optimization of tack scheduling in cloud environments. They formulated the objective function based on cloud throughout, utilization of resources and imbalance Degree. S. Pang et al. [32] developed a method called as "Estimation of distribution algorithm (EDA)" and used GA for scheduling tasks in cloud technology. Initially, they generated the probability model and sampling mechanism of EDA to create the feasible solutions' scalability. Next, they applied crossover operation followed by mutation to expand the range of solutions. Boonhatai Kruekaew [33] used ABC and Q-learning algorithms for task scheduling based on multiple objectives. The Q-leaning is a reinforcement algorithm and employed to fasten the ABC algorithm. They considered maximizing the throughput of VMs, optimizing the resource utilization, and balancing the load between VMs. Priya and Babu [34] proposed a "Moving average fuzzy resource scheduling (MAFRS)" for cloud virtual centers which are located in virtual fashion. They designed a fuzzy control theory for system accessibility between cloud user's resources and cloud requirements. Karanam Sreenu and M. Sreelatha [35-37] employed three Meta-Heuristic algorithms namely "Whale Optimization algorithm (WOA)", "Fractional GWO (FGWO)" and "Modified fractional GWO (MFGWO)" for efficient task scheduling. FGWO and MFGWO used resource utilization, energy, communication cost, Execution cost and communication time as reference parameters for task scheduling. WOA used makespan and budget cost for fitness function formulation.

In summary, the Meta-heuristic-based task scheduling solves the convergence problem effectively. But they have induced huge computational burden at the scheduler. The complexity reaches to worst in the case of more number of requests from multiple users. In such case, the simple task scheduling is required which inevitable allocates the resources for all users without any delay.

3. Proposed approach

3.1 Overview

Here, we develop a new task Scheduling scheme for the cloud environments. The proposed model mainly considers four parameters at the task scheduling process for resource selection; they are resource reaction time, resource location, resource availability and resource reliability rate. Based on these four parameters, the CSP assigns the tasks to resources. Initially, all the parameters are calculated individually and then one index called as task scheduling index (TSI) is measured. Based on the TSI values, optimal resources are selected and the tasks are assigned. Here the main intention behind the consideration of four parameters is to make the cloud computing effective and resilient from several resource constraints. Since the resources participating in cloud are of constrained to several issues, we referred to view each resource in multiple orientations and then selected those have optimal performance. Among the proposed four parameters, three shows the effectiveness at the Quality-of-service provision and one is purely for reliability assessment. Since both are important for cloud environments, we have referred four parameters to assess the resource efficiency.

3.2 Methodology

In this section, we explain the details of proposed task scheduling system that employs an adaptive resource aware task scheduling in cloud environments. Fig. 1 shows the simple schematic of proposed scheme which illustrates the interaction between different cloud components from cloud User's task submission to task completion. Majorly, the proposed method composed of four components; they are cloud user (CU), cloud task scheduler (CTS),

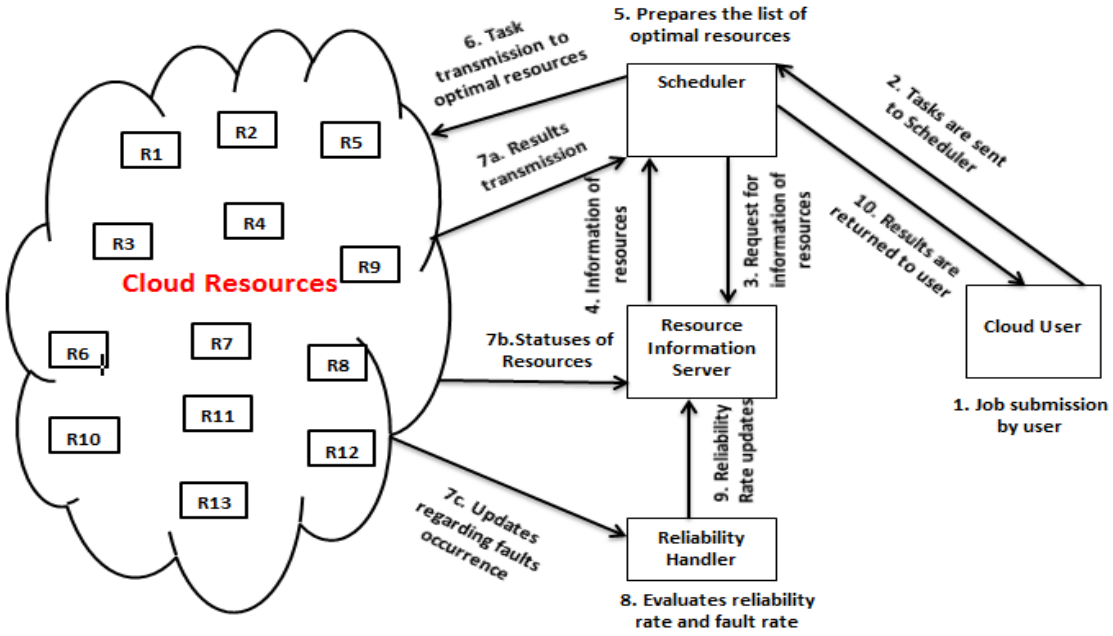

Figure. 1 working flow of proposed task scheduling mechanism

cloud resource information server (RIS) and Reliability Handler (RH). The details of individual components are illustrated in the following sub-sections;

3.2.1. Cloud user

In the initial phase, the cloud users put request to cloud scheduler mentioning that the user wants to accomplish its task through the cloud resources. Tasks sent by the cloud user are received by Cloud Task scheduler. Once the tasks are received at cloud scheduler, it contacts the RIS which has the entire information about the cloud resources. RIS provides the information about the resource to CTS. Then the CTS measure the optimality of resources through TSI and the resource those have an optimal TSI are only selected for tasks execution. Once the resources are selected, the tasks at CTS are dispatched to them. After the completion of task, the results are returned to the respective cloud user. RH is responsible for the detection of faults of resources and predicting the resource's reliability rate.

3.2.2. Resource reaction time

Reaction Time is the most important parameter through which the resource's timely execution can be predicted. The reaction time can be estimated from the time taken by resources and CTS to complete the task. For a given resource, if it was busy with some others task, then it can't complete the task quickly. Hence, we have considered the reaction time as one of the parameters for the selection of resources. Generally, the reaction time considers the limits of time at which the cloud user had submitted the task and the time at which the cloud user had received the results. Along with these two-time instances, it also considers the time take by resource to execute the task. Hence the reaction time of a resource is calculated as the sum of three individual times; they are (1) The time taken to transmit the task from CTS to resource, (2) The time taken by resource to execute the task and (3) The time taken to get the results back from resource to CTS. Mathematically, the reaction time of resource a for a task b is expressed as

$$RT_a^b = \sum_{p \in s,e,r} T_{pb} \quad (1)$$

Where, RT_a^b is the reaction time of resource a for a task b, T_{sb} is the time taken by scheduler to transmit the task b to resource, T_{eb} is the time taken by resource a to execute the task b and T_{rb} is the time taken to receive the results from resource a to CTS. T_{sb} is defined as

$$T_{sb} = \frac{L_{sb}}{\omega_a} \quad (2)$$

Where, L_{sb} is the size of task b and ω_a is the bandwidth assigned for the channel between resource a and CTS. Next, T_{eb} is defined as,

$$T_{eb} = \frac{E_b}{v_a} \quad (3)$$

Where, E_b is the time taken by resource a to execute the task b and v_a is the speed of resource a. Next T_{rb} is defined as

$$T_{rb} = \frac{L_{rb}}{\omega_a} \quad (4)$$

Where, L_{rb} is the size of results to be transmitted by resource to CTS. In the reaction time, the T_{sb} and T_{rb} are purely depends on the size and bandwidths assigned to resources. As the Bandwidth between CTS and resource increases, the reaction time decreases and vice versa. Further, with an increase in the size of task, the reaction time also increases. Hence we have modeled the reaction time with respect to size of task and bandwidth of resources.

3.2.3. Resource location

In cloud computing, the resources are much important to execute the task with less failure rate. The failure rate is related to the location of resources. For a static resource (resource available for long time in a single location), the probable failure rate is less and for a dynamic resource (resource dynamically changing the locations and not available for longer durations in a single location), the probable failure rate is high. Hence the resource location is important and we consider it as one of resource selection parameter. Depends on the location of resources, we have divided them into two categories; they are neighbor resources and distant resources. The neighbor resources are the ones which work under the cloud maintainer and are located close to the cloud maintainer. On the other hand, the distant resources are the ones which work temporarily and are located far from the cloud maintainer. Distant resources are connected through communication links, and they are bound to certain policies and configurations of which they are the part. A change in the configuration or policy of the network of distant resources may raise the question about the accessibility and availability of resources. The distant resources are chosen only if they have unique property. Otherwise, if it looks like a common resource, then the location has to consider during the selection of resources. Moreover, the situation becomes worse if the communication link breaks down due to several factors which are not under the control of cloud maintainer. Hence the locally available resources are always showing an efficient performance when compared with the distant resources.

Here the categorization of resources is done based on the calculation of Euclidean distance between cloud maintainer and resources. For a given cloud maintainer, this paper allocates a fixed communication range. After the computation of Euclidean distances between cloud maintainer and resources, the categorization is done through the following expression.

$$R_i = \begin{cases} L_{R_i}, & if \ \frac{|d(CM,R_i)|}{\max(d)} \leq \delta \\ D_{R_i}, & Otherwise \end{cases} \quad (5)$$

Where, L_{R_i} is the neighbor resource which is available locally and D_{R_i} is the distant resource through which a communication link exists with cloud maintainer. δ is the communication range of Cloud maintainer. For a given locations of cloud maintainer and the i^{th} resource, the Euclidean distance is measured as

$$d(CM, R_i) = \sqrt{(CM(x) - R_i(x))^2 + (CM(y) - R_i(y))^2} \quad (6)$$

Where, $(CM(x,y))$ and $R_i(x,y)$ are the location coordinates of Cloud maintainer and i^{th} resource respectively. Here we have considered the normalized Euclidean distance and the communication range is considered to be in the range of 0 to 1. From the expression shown in Eq. (5), we can say that the resources those have Euclidean distance less than or equal to the communication range of cloud maintainer are called as neighbor resources and otherwise they are called as distant resources. Based on this analysis, we can say the neighbor resource have more weightage than the distant resources. At worst conditions means there is no availability of even a single neighbor resource, the distant resources are selected. At this phase, again employ the same Euclidean distance-based selection for distant resources selection. The distant resources are selected those are close to cloud maintainer, according to the following expression.

$$SR_i = \min_i [d(CM, D_{R_i})] \quad (7)$$

Where, SR_i is the selected distant resource that has minimum Euclidean distance with cloud maintainer.

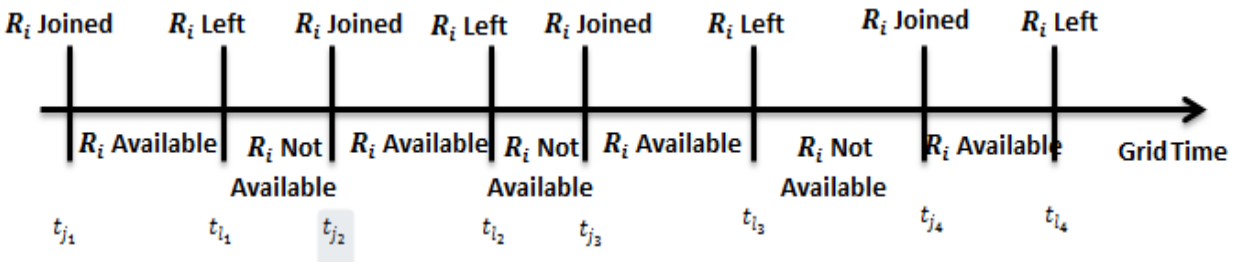

Figure. 2 Resource availability and unavailability

3.2.4. Resource availability

In cloud computing, the resources at various geographical locations can participate to execute the tasks assigned by CTS. Due to different geographical locations, the resources cannot stay fixed for longer time and they have tendency to leave or join the cloud [38]. If any resource has left without any prior information, then it causes a serious issue. Hence it need to be consider the tendency of joining and leaving times of resources to and from respectively. This is formulated with respect to the component called as resource availability. The resource availability is defined as the time from which it was joined to the cloud. Similarly, the unavailability of a resource is defined with respect to the time from which the resource is not available.

For a given resource, both the availability and unavailability have multiple instances. For example, consider a resource is joined at time instance t_1 and left at time instance t_2. Again, the same resource is joined to the cloud network and let it be time instance t_3 and again left at time instance t_4. In this example, we can observe that there are two available instances and two unavailable instances. The time difference between t_2 and t_1 can be defined as the available time and the time difference between t_3 and t_2 can be considered as unavailable time. Similarly, the time difference between t_4 and t_3 gives the second available time. Hence we consider the multiple instances and derived the mean times based on them. Consider a resource joined the network for five times such as t_{j_1}, t_{j_2}, t_{j_3}, t_{j_4} and t_{j_5} and also there are five unavailable times, such as t_{l_1}, t_{l_2}, t_{l_3}, t_{l_4} and t_{l_5}. For both times, we compute the mean time and they are mathematically expressed as

$$A_T = \sum_{i=1}^{N} \frac{\left(t_{l_i} - t_{j_i}\right)}{N} \tag{8}$$

and

$$\hat{A}_T = \sum_{i=1}^{N} \frac{\left(t_{j_{i+1}} - t_{l_i}\right)}{N} \tag{9}$$

Where, A_T and $\hat{A}_T$ are Mean Available Time and Mean unavailable time respectively. In the Eq. (8), the term $\left(t_{l_i} - t_{j_i}\right)$ calculates the available time of a resource and in the Eq. (9), the term $\left(t_{j_{i+1}} - t_{l_i}\right)$ calculates the unavailable time of resources. Further the terms t_{j_i}, t_{l_i} and $t_{j_{i+1}}$ denotes the i^{th} instant of joining time, i^{th} instant of leaving time and $i+1^{th}$ joining times respectively. Here to compute the available time, we have simply subtracted the i^{th} joining time form i^{th} leaving time. Next for unavailability computation, the i^{th} leaving time is subtracted from $i+1^{th}$ joining time. Based on these two A_T and $\hat{A}_T$, we compute two factors called as Availability factor (A_i) and unavailability factor (UA_i), as

$$A_i = \frac{A_T}{A_T + \hat{A}_T} \tag{10}$$

and

$$UA_i = \frac{\hat{A}_T}{A_T + \hat{A}_T} \tag{11}$$

Based on these two factors, the resource availability is measured. A_i is preferred to be high which denotes a longer availability time and UA_i is preferred to be low which denotes a shorter unavailability time. For the execution of assigned tasks, the longer time availability gives more efficiency because the tasks are of different sizes, means some tasks are smaller is size and some tasks are larger in size. To execute the former type of tasks, the less time availability is enough while for second type of tasks, the longer time availability is required. Furthermore, as the availability of resource increases, the CTS assign the tasks for multiple times which in turn can provide a rich income for the resource provider. Fig. 2 explores the concept of resource availability.

As shown in the above Fig. 2, the resource R_i has joined totally for four times and also left for four times. In this context, the A_T is calculated as $A_T =$

$(t_{l_1} - t_{j_1}) + (t_{l_2} - t_{j_2}) + (t_{l_3} - t_{j_3}) + (t_{l_4} - t_{j_4})/4$. Similarly, the $\hat{A}_T$ is calculated as $\hat{A}_T = (t_{j_2} - t_{l_1}) + (t_{j_3} - t_{l_2}) + (t_{j_4} - t_{l_3})/3$. Based on these two instances, the availability and unavailability of a resource R_i is measured and its probability of selection is estimated. If it found that the resource has high availability, then its probability to get selected is high otherwise, it can't be selected.

3.2.5. Resource reliability rate

Reliability rate is the key component in cloud computing, because lower reliability rates of any resource will consequences to so many problems. The reliability rate is defined here with respect to fault rate. If any resource failed at the mid of task execution, then there may be a huge information loss and also there is a chance of information misuse. Hence the fault rate is needed to be considered during the selection of resources in computational clouds. According to the fault rate, a resource is selected which has less fault rate which is considered as highly reliable resource.

In general, most of the earlier approaches employed fault index [39-42] for the selection of a reliable resource. The fault index is done according to the history information of fault tolerance. The fault index is increased every time when the resource doesnot complete the task successfully within the given time deadline. Similarly, the fault index is decremented every time when the resource complete the task successfully within the given time deadline. So the optimal resource selection is done whenever the fault index reaches to zero which is not practically possible. Since the resource in cloud computing has so many constraints, no resource can achieve the fault index as zero. Network failure, hardware failures and prediction failures are some of the examples of failures those occur frequently in computational cloud networks. Hence we have derived a new component called as reliability rate which is measured based on the history of successes occurred during the execution of tasks. The resources those have higher reliability rate are only selected to execute the tasks and the resources those have higher fault rate are not selected. Mathematically, the reliability rate is measured as

$$R_r = \frac{S_N}{S_N + F_N} \text{ or } F_r = \frac{F_N}{S_N + F_N} \quad (12)$$

Here R_r is reliability rate and F_r is fault rate which are related with inverse proportion. F_N is defined as the overall failed instances and S_N is defined as the overall succeeded instances. As the F_N value increases, the F_r also increases and as the S_N increases, the R_r increases. For an optimal resource, the fault rate must be low and reliability rate must be high. The values of F_N and S_N are maintained by the cloud information server and they are fed to cloud scheduler whenever it required. For every instance of success, the S_N value is increased and for every instance of failure, the F_N value increases and they are updated to RIS.

3.2.6. Resource selection matrix

Finally to get the optimal resources, we have constructed a resource selection matrix based on the above specified four parameters.

In the above Table. 1, we assign less weightage for the resources those have no reliability, means if the reliability is '0', the values of location, reaction time and availability are ignored to that resource and it was disqualified from the list of available resources at CTS. For that combination, the TSI is assigned as 0. Next, the weightage is given for the resources those have any three combinations produce '1'. For such kind of resource the TSI is calculated as the ratio of sum of current total ones to the maximum possible sum of ones (i.e., ¾ = 0.75). Next, the weightage is given for the resources those have any two combinations produce '1'. For such kind of resource the TSI is calculated as the ratio of sum of current total ones to the maximum possible sum of ones (i.e., 2/4 = 0.50). Next, the weightage is given for the resources those have any one combinations produce '1'. For such kind of resource, the TSI is calculated as the ratio of sum of current total ones to the maximum possible sum of ones (i.e., 1/4 = 0.25). Finally the least weightage is given for the resources those have all '0''s and higher weightage (i.e., 1) for the resources those have all '1's.

4. Experimental results

In this section, we explore the details of simulation experiments conducted over the developed task scheduling model. We have conducted a vast set of experiments to validate the developed model with varying parameters like number of tasks submitted. By varying these parameters, we compute the performance through several performance metrics such as throughput (tasks/sec), Makespan (sec) and convergence time (Sec). At every phase of simulation, the obtained performance values of developed method are compared with several existing methods such as GA [32], PSO [25], and GWO [31]. Initially, we explore

Table. 1 resource selection matrix

Combination	Reaction Time	Location	Availability	Reliability Rate	TSI
X1	0	0	0	0	0
X2	0	0	0	1	0.25
X3	0	0	1	0	0
X4	0	0	1	1	0.5
X5	0	1	0	0	0
X6	0	1	0	1	0.5
X7	0	1	1	0	0
X8	0	1	1	1	0.75
X9	1	0	0	0	0
X10	1	0	0	1	0.5
X11	1	0	1	0	0
X12	1	0	1	1	0.75
X13	1	1	0	0	0
X14	1	1	0	1	0.75
X15	1	1	1	0	0
X16	1	1	1	1	1

the details of datasets used, next the simulation results.

4.1 Datasets

For experimental validation, we totally used two standard datasets namely, "Google cloud jobs (GoCJ) dataset" [43], and 3) "Synthetic workload dataset" [44], which are described as:

4.1.1. GOCJ dataset

This is treated like Realistic Google dataset and was made from the workload behaviors happened in the traces of Google Cluster. A well-known simulation method called as Monte Carlo simulation is employed for this dataset creation. The size of tasks in GOCJ dataset ranges from 15k to 900k Million Instructions (MIs). Totally the tasks in this dataset are classified into four categories; they are small sized tasks, medium sized tasks, larger sized tasks, extra larger sized tasks and huge larger sized tasks ranging from 15k-55k MIs, 59k-99k MIs, 101k-135k MIs, 150k-337.5k MIs and 525k-900k MIs respectively. This dataset is provided for the evaluation purposes in the form of different text files in the Mendeley repository. The data is organized in different rows and columns which consist of numeric values. Each numerical value signifies the cloud task size through MIs.

4.1.2. Synthetic workload dataset

This dataset is generated using the two methods, they are Monte carlo simulation and a random number generator mechanism. The size of tasks in this dataset ranges from 1 to 45k million instructions (MIs). Totally the tasks in this dataset are classified into five categories; they are tiny sized tasks (1-250 Mis), small sized tasks (800-1200 Mis), medium sized tasks (1800-2500 Mis), larger sized tasks (7k-10l Mis), and extra larger sized tasks (30k-45k Mis).

4.2 Results

Under the results evaluation, we referred three performance metric assesses the performance of proposed approach. The three metrics are namely Makespan, Throughput and Convergence in terms of iterations. These metrics are measured with varying number of tasks submitted to the cloud. Further, at every validation, the performance of proposed method is compared with several existing methods such as GA [32], PSO [25] and GWO [31]. The detailed interpretation about the performance metrics is given below.

4.2.1. Makespan

Makespan is one of the mostly used performance measure used in the cloud computing for the assessment of performance in terms of task completion time. The method is said to be better if it achieves less Makespan and vice versa. In our experiments, the Makespan is measured for two times one time for GOCJ dataset and another time for Synthetic dataset.

Fig.3 and Fig. 4 shows the performance of proposed approach in terms of Makespan for varying number of tasks submitted for GOCJ dataset and Synthetic dataset respectively. As it can be seen, the Makespan increases for an increasing nature of Tasks submitted. For a contrast set of resources in cloud, with an increase in the number of tasks submitted to cloud, the resources put the tasks in Queue and hence

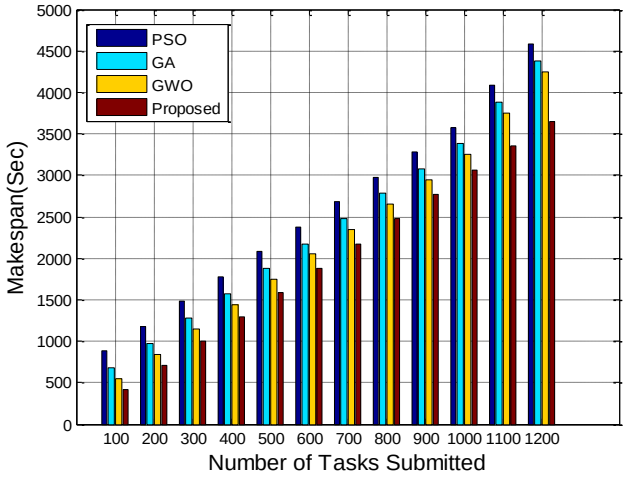

Figure. 3 Makespan for varying number of tasks for GOCJ dataset

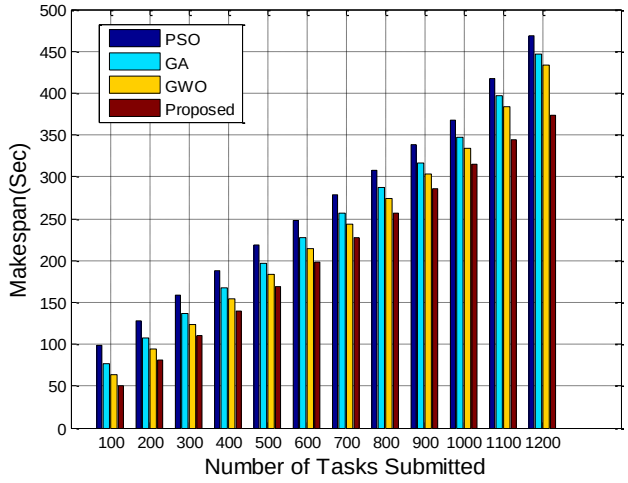

Figure. 4 Makespan for varying number of tasks for synthetic workload dataset

the Makespan increase. However, the proposed method is observed to have less Makespan as it follows a simple iterative mechanism for task scheduling. Our method works based on the conditional logic which takes very less time to assign a task for resources. In the case of earlier Meta-heuristic algorithms, they formulate an objective function and optimize it iteratively.

This process takes huge time even for small number of tasks. From Fig. 3, we can see that the range of Makespan is observed as 0 – 4700 seconds. Next, from Fig.4, we can see that the range of Makespan is observed as 0 – 480 Seconds. As the text files of GOCJ dataset are composed of huge number of Instructions, they need more time for task scheduling as well as for execution. Compared to GOCJ, the text files of synthetic dataset has only 5% instruction, they consumed very less time for execution. Hence the range of Makespan is very less

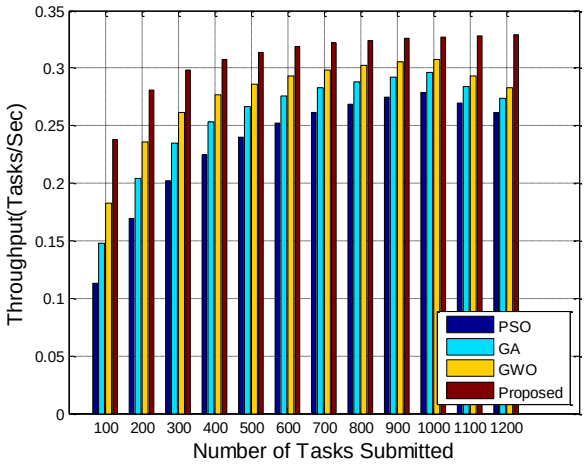

Figure. 5 Throughput for varying number of tasks for GOCJ dataset

for synthetic dataset compared to GOCJ dataset. On an average, the Makespan of proposed method for GOCJ dataset is observed as 1558 seconds while for PSO, GA and GWO, it is observed as 2020 Seconds, 2340 seconds and 2568 seconds respectively. Similarly, the average Makespan of proposed method for Synthetic workload dataset it is observed as 160 seconds while for PSO, GA and GWO, it is observed as 205 Seconds, 232 seconds and 268 seconds respectively.

4.2.2. Throughput

Throughput is one more popular measure used to verify the effectiveness of task scheduling methods. Throughput is measured as the total number of tasks completed per second. The method is said to be better if it achieves larger Throughput and vice versa. In our experiments, the Throughput is measured for two times one time for GOCJ dataset and another time for synthetic dataset.

Fig. 5 and Fig. 6 shows the performance of proposed approach in terms of Throughput for varying number of tasks submitted for GOCJ dataset and synthetic dataset respectively. Throughput has a indirect relation with makespan, as the makespan is less then the throughput is high and vice versa. For a cloud service provider, if the overall count of completed tasks within the less timespan are more, then the throughout will be better. So, for less Makespan the throughout is high. From the figures, we can see that the the througput increases with an increase in the number of tasks submitted to the cloud. Since the proposed method consumes very less time in the selection of resources for task scheduling, it can complete more number of tasks and hence the propsoed method has higher throughput.

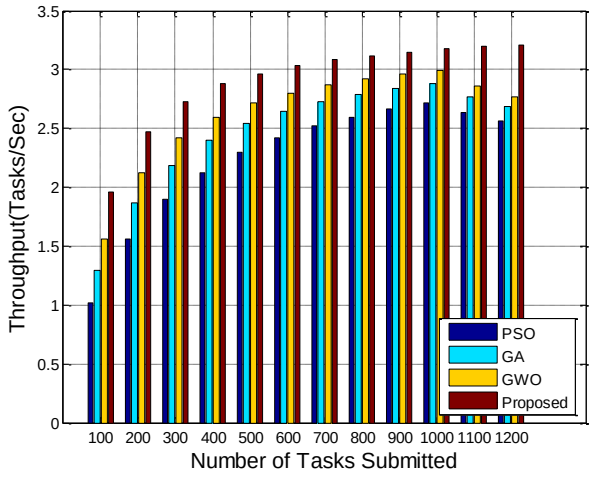

Figure. 6 Throughput for varying number of tasks for synthetic workload dataset

Figure. 7 Convergence for varying number of tasks

Compared to the iteratve optimization, the conditional logic based optimiation lessens the computational time much effectively and makes the system to execute more number of tasks even in the less timespan. As the earlier meta-heuristic algorithms commonly follows an iterative optimization, they had experienced more Makespan consequently less throughput. On an average, the Throughput of proposed method for GOCJ dataset is observed as 0.3562 tasks/sec while for PSO, GA and GWO, it is observed as 0.2325 tasks/sec, 0.2540 tasks/sec and 0.2785 tasks/sec respectively. Similarly, the average Throughput of proposed method for Synthetic workload dataset it is observed as 3.5210 tasks/sec while for PSO, GA and GWO, it is observed as 2.2520 tasks/sec, 2.4650 tasks/sec and 2.6330 tasks/sec respectively.

4.2.3. Convergence speed

Convergence speed measures the total number of iterations in which the task scheduling mechanism has find out an optimal resources for task assigning. As the numbers of iterations taken to get optimal resources set are more, the convergence is said to be slow and vice versa. If the task scheduling time is reduced, then the CSP can provide more number of services for more user requests. With this aim, we proposed a simple task scheduling mechanism based on multiple resource attributes and conditional logic. To verify the convergence of proposed method applied to task scheduling in cloud computing, we compared to it with the earlier Meta-heuristic algorithms such as GA, PSO and GWO. Fig. 7 shows the convergence achieved with varying number of tasks submitted to cloud. As we can see from this Figure, compared to other methods, the proposed method required very less number of iterations to complete the convergence. Since the proposed method selects optimal resources based on a simple conditional logic, it has experienced faster convergence speed. Among the earlier methods, the faster convergence is achieved for GWO and slower convergence is achieved for PSO. As PSO needs to perform local as well as global optimization, it needs more number of iterations to get converged. Additionally, with the rise in the tasks to get executed, the convergence becomes slow and the system will consume more time to complete the bulk data. On an average, the convergence of proposed method is observed as 194 iterations while for PSO, GA and GWO, it is observed as 270, 254 and 230 respectively.

5. Conclusion

Cloud computing is a large scale parallel and distributed computing system which composed of huge number of cloud resources and communication links. However, the proper and balanced task execution by obtaining an optimal virtual machine is a key challenge in cloud computing. Aiming at the lower reliability and imbalanced task scheduling, this paper proposed a simple and effective task scheduling mechanism based on the multiple resources attributes and conditional logic. This method considered totally four resource attributes such as reaction time, availability, location, and reliability to formulate the conditional logic. Based on the resource attributes, a task scheduling index is calculated to find the optimal resources of task scheduling with three objectives of minimum Makespan, maximum Throughput and faster convergence. Simulation experiments are conducted and the performance of proposed method is compared with several state-of-the-art Meta-heuristic

algorithms such as GA, PSO and GWO. This strategy showed an outstanding performance in reducing the Makespan, maximizing throughput and fastening the convergence speed. On an average, the improved throughput of proposed method is 28.40% and 19.90% for GOCJ and Synthetic datasets respectively.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, Design, Development, and implementation of proposed task scheduling by K. Sreenu and Validation and proofread by M. Sreelatha.

References

[1] J. Lee, "A view of cloud computing", *Int. J. Netw. Distrib. Comput.*, Vol. 1, No. 1, pp. 2-8, 2013.

[2] M. Ibrahim, "SIM-cumulus: A large-scale network-simulation-as-aservice", *Ph.D. dissertation, Dept. Comput. Sci., Capital Univ. Sci. Technol*., Islamabad, Pakistan, 2019.

[3] N. J. Navimipour, A. M. Rahmani, A. H. Navin, and M. Hosseinzadeh, "Expert cloud: a cloud-based framework to share the knowledge and skills of human resources", *Comput. Hum. Behav*. Vol. 46, pp. 57–74, 2015.

[4] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds", *Future Gener. Comput. Syst*., Vol. 48, pp. 1–18, 2015.

[5] N. J. Navimipour, "A formal approach for the specification and verification of a trustworthy human resource discovery mechanism in the expert cloud", *Expert Syst. Appl.*, Vol. 42, No. 15, pp. 6112–6131, 2015.

[6] E. N. Alkhanak, S. P. Lee, and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: taxonomy and opportunities", *Future Gener. Comput. Syst*., Vol. 50, pp. 3–21, 2015.

[7] S. Yang, L. Pan, Q. Wang, S. Liu, and S. Zhang, "Subscription or pay-as-you-go: Optimally purchasing IaaS instances in public clouds", In: *Proc. of IEEE Int. Conf. Web Services (ICWS)*, San Francisco, pp. 219-226, 2018.

[8] A. Sajjad, A. A. Khan, and M. Aleem, "Energy-aware cloud computing simulators: A state of the art survey", *Int. J. Appl. Math., Electron. Comput.*, Vol. 6, No. 2, pp. 15-20, 2018.

[9] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. D. Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", *Softw., Pract. Exper.*, Vol. 41, No. 1, pp. 23-50, 2011.

[10] M. Masdari, S. V. Kardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis", *J. Netw. Comput. Appl.*, Vol. 66, pp. 64–82, 2016.

[11] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey", *Future Generation Computer Systems*, Vol. 91, pp. 407–415, 2019.

[12] V. Hayyolalam and A. A. Kazem, "A systematic literature review on QoS-aware service composition and selection in cloud environment", *J. Netw. Comput. Appl.*, Vol. 110, pp. 52–74, 2018.

[13] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. Zhou, "A task scheduling algorithm based on QoS -driven in cloud computing", *Comput. Sci.*, Vol. 17, pp. 1162–1169, 2013.

[14] R. Albodour, A. James, and N. Yaacob, "QoS within business cloud quality of service (BGQoS)", *Future Gener. Comput., Syst.*, Vol. 50, pp. 22–37, 2015.

[15] H. E. D. Ali, I. A. Saroit, and A. M. Kotb, "Grouped tasks scheduling algorithm based on QoS in cloud computing network", *Egyptian Inform. J.*, Vol. 18, No. 1, pp. 11–19, 2017.

[16] M. S. Thampi, E. Sayed M. E. AIfy and L. Trajkovic, "Cost-enabled QoS aware task scheduling in the cloud management system", *Journal of Intelligent and Fuzzy Systems*, Vol. 41, No. 5, pp. 5607-5615, 2021.

[17] B. A. A. Maytami, "A Task Scheduling Algorithm With Improved Makespan Based on Prediction of Tasks Computation Time algorithm for Cloud Computing", *IEEE Access*, Vol. 7, pp. 160916-160926, 2019.

[18] H. Mahoumd, "Multi-objective Task Scheduling in Cloud Environment Using Decision Tree Algorithm", *IEEE Access*, Vol. 10, pp. 36140-36151, 2022.

[19] Q. Guo, "Task Scheduling Based on Ant Colony Optimization in Cloud Environment", In: *Proc., of AIP Conference*, Poland, pp. 0400391-11, 2017.

[20] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing", *Journal of*

Ambient Intelligence and Humanized Computing, Vol. x, pp. 1-13, 2020.

[21] G. Li and Z. Wu, "Ant Colony Optimization Task Scheduling Algorithm for SWIM Based on Load Balancing", *Future Internet*, Vol. 11, p. 90, 2019.

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

[23] E. Elsedimy, and F. Algarni, "MOTS-ACO: An improved ant colony optimizer for multi-objective task scheduling optimization problem in cloud data centers", *IET Networks*, Vol. 11, No. 2, pp. 43-57, 2021

[24] N. Dordaie, N. J. Navimipour, "A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments", *J. ICT Exp*., Vol. 4, No. 4, pp. 199-202, 2017.

[25] X. Huang, C. Li, H. Chen, and A. Dong, "Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies", *Cluster Computing*, Vol. 23, pp. 1137–1147, 2020.

[26] H. M. Eldesokey, S. M. A. E. Atty, W. E. Shafai, M. Amoon, and F. E. A. E. Samie "Hybrid swarm optimization algorithm based on task scheduling in a cloud environment", *International Journal of Communication Systems*, Vol. 34, No. 13, pp. 37-47, 2021.

[27] Z. Wu and J. Xiong, "A Novel Task Scheduling algorithm of cloud computing based on Particle Swarm Optimization", *International Journal of Gaming and Computer-Mediated Simulations*, Vol. 13, No. 2, pp. 1-15, 2021.

[28] Sharma, S. Kumar, and Kumar, Nagresh, "A Modified Particle Swarm Optimization for Task Scheduling in Cloud Computing", In: *Proc. of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE)*, Tirunelveli, Tamilnadu, pp. 176-181, 2019.

[29] 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.

[30] M. Sudheer, and M. V. Krishna, "Multi Objective Task Scheduling Algorithm in Cloud Computing Using the Hybridization of Particle Swarm Optimization and Cuckoo Search", *Journal of Computational and Theoretical Nanoscience*, Vol. 17, No. 12, pp. 5346-5357, 2020.

[31] D. Alsadie, "TSMGWO: Optimizing Task Schedule Using Multi-Objectives Grey Wolf Optimizer for Cloud Data Centers", *IEEE Access*, Vol. 9, pp. 37707-37725, 2021.

[32] S. Pang, W. Li, H. He, Z. Shan, and X. Wang, "An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing", *IEEE Access*, Vol. 7, pp. 146379-146389, 2019.

[33] B. Kruekaew, and W. Kimpan "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning", *IEEE Access*, Vol. 10, pp. 1783-17818, 2022.

[34] V. Priya and C. N. K. Babu, "Moving average fuzzy resource scheduling for virtualized cloud data services", *J. Comput. Standards Interfaces*, Vol. 50, pp. 251–257, 2018.

[35] K. Sreenu and M. Sreelatha, "W-Scheduler: whale optimization for task scheduling in cloud computing", *Cluster Comput.*, Vol. 22, pp. 1087-1098, 2017.

[36] K. Sreenu and M. Sreelatha, "FGMTS: Fractional grey wolf optimizer for multi-objective task scheduling strategy in cloud computing", *Journal of Intelligent & Fuzzy Systems*, Vol. 35 pp. 831–844, 2018.

[37] K. Sreenu and M. Sreelatha, "MFGMTS: Epsilon Constraint-Based Modified Fractional Grey Wolf Optimizer for Multi-Objective Task Scheduling in Cloud Computing", *IETE Journal of Research*, Vol. 65, No. 2, pp. 201-215, 2018.

[38] P. Latchoumy and P. S. A. Khader, "A combined approach: Proactive and reactive failure handling for efficient job execution in computational grid", In: *Proc. of Int. Comput. Netw. Inf.*, India, pp. 713-725, 2014.

[39] K. Srinivasa, G. Siddesh, and S. Cherian, "Fault-tolerant middleware for grid computing", In: *Proc. of 12th IEEE International Conference on High Performance Computing and Communications*, Melbourne, Australia, pp. 635–40, 2010.

[40] R. S. Chang, J. S. Chang, and P. S. Lin, "An ant algorithm for balanced job scheduling in grids", *J. Future Gen. Comput. Syst.*, Vol. 25, pp. 20-27, 2009.

[41] L. Chunlin, Z. J. Xiu, and L. Layuan, "Resource scheduling with conflicting objectives in grid environments: model and evaluation", *J Netw Comput Appl.*, Vol. 32, No. 3, pp. 760–769, 2009.

[42] P. Huang, H. Peng, P. Lin, and X. Li, "Static strategy and dynamic adjustment: an effective method for grid task scheduling", *J Future Gen Comput Syst,*, Vol. 25, No. 8, pp. 884–92, 2009.

[43] A. Hussain and M. Aleem, "GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures", *Data*, Vol. 3, No. 4, p. 38, 2018.
[44] A. Hussain, M. Aleem, A. Khan, M. A. Iqbal, and M. A. Islam, "RALBA: A computation-aware load balancing scheduler for cloud computing", *Cluster Comput.*, Vol. 21, No. 3, pp. 1667-1680, 2018.