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

A Novel Modified and Optimized Meta-Heuristic Load-Balancing Technique for Cloud Computing System

D. Chandrasekhar Rao[1,*], Suraj Sharma[2], Sanjib Kumar Nayak[3], Suresh Kumar Srichandan[4], Alina Dash[5]

Submitted: 25/04/2023 **Revised**: 28/06/2023 **Accepted**: 06/07/2023

Abstract: Cloud computing has emerged as a transformative technology that offers vast computational resources to meet the growing demands of modern applications. However, the efficient allocation of these resources to ensure optimal performance and scalability remains a critical challenge. Load balancing techniques play a pivotal role in optimizing resource utilization and improving the overall performance of cloud-based systems. Cloud service providers are looking for creative ways for dispersing the load across the virtual machines. Recent study suggests that efficient task scheduling or task-virtual machine mapping techniques can be used to achieve load balancing. It's also a well-known NP-Hard problem. Hence, contrary to polynomial-time algorithms, the researchers have been searching for meta-heuristic algorithms. In order to provide a solution to the mentioned issue, this research introduced a modified firefly swarm algorithm. The primary goal is to meet all deadlines while reducing the total amount of time it takes to execute all tasks. The proposed technique is compared to particle swarm optimization, bacteria foraging optimization, and dragonfly optimization to demonstrate its efficacy.

Keywords- *load balancing; task scheduling; optimization; cloud computing; virtual machine*

1. Introduction

In cloud computing, load balancing occurs when a user's request is matched with available virtual machines. Hence, cloud computing relies heavily on job scheduling. Two types of task scheduling procedures can be distinguished: static and dynamic. In static task scheduling, all user tasks are queued up at the start, and the virtual machine schedules them in a static fashion. In contrast, the dynamic scheduling mechanism maps the job to the virtual machine at runtime, whenever the user's task enters the system. Algorithms for scheduling tasks are used for the effective administration of virtual machines and task processing. Initially, customers and cloud service providers would sign a Service Level Agreement (SLA) when users submitted tasks for processing to the cloud system. Quality of service (QoS) requirements pertaining to job scheduling are specified in the Service Level Agreemen (SLA). There is a well-defined set of quality-of-service standards, including a set budget, a set timeframe within which tasks must be completed, and a set of security-related services. Users are billed based on how much they utilize the cloud's resources, as cloud services operate on a "pay as you go" basis. The user needs to know how many virtual machines will be required for each operation, and they should factor that into their budget. When it comes to cloud computing, scheduling tasks is a non-deterministic NP-Hard problem that can be thought of as the bin-packing problem. As cloud computing becomes more complicated, the NP-hard problem gets harder to solve. The optimal solution for the scheduling problem in a short span of time can be obtained by meta-heuristic algorithms. Practical Swarm Optimization (PSO) and Genetic Algorithm (GA), have been used in recent years to solve the cloud scheduling problem, yielding near-optimal outcomes in a shorter amount of time. Researchers pay close attention to the outcomes produced by metaheuristic algorithms, leading to numerous algorithm proposals for task scheduling and load balancing. To provide the appropriate solution for the above-mentioned problem, in this paper, we proposed a solution by using Fireflies Swarm Optimization (FSO) algorithm to improve the execution cost for the user task submitted into the cloud infrastructure. The cost of execution would be determined by adding together the time it takes to do the operation and the price at which cloud resources, like virtual computers, are used. The task's completion time is calculated by combining the execution and waiting times. The completion time should be less than or equal to the deadline defined by the user at the time of task submission. The effectiveness of the proposed Fireflies Swarm Optimisation Task Scheduling (FSOTS) algorithm for task scheduling was examined through a

[1,4]Department of Information Technology, Veer Surendra Sai University of Technology, Burla, Odisha
[2]Department of Computer Science & Engineering, Guru Ghasidas Vishwavidyalaya, Raipur, Chhatisgarh,
[3]Department of Computer Application, Veer Surendra Sai University of Technology, Burla, Odisha
[5]Department of Computer Science & Engineering, Veer Surendra Sai University of Technology, Burla, Odisha
Email:[1]dcrao_it@vssut.ac.in, [2]suraj.cse@ggu.ac.in, [3]sknayak_ca@vssut.ac.in, [4]suresh_it@vssut.ac.in, [5]alinadash_cse@vssut.ac.in
*(*Corresponding author's e-mail: dcrao_it@vssut.ac.in)*

series of simulation experiments, and the outcomes were contrasted with those of other metaheuristic algorithms like PSO, Dragon Fly Optimisation (DFO), and Bacteria Foraging Algorithm (BFA).

The remaining sections are organized as follows: Existing relevant works have been discussed in Section II. The task scheduling problem is formulated in Section III, Section IV elaborates the proposed methodology for problem solution, control parameters for simulation and result analysis are presented in Section V, and the study is concluded in Section VI with future scope.

2. Related Work

Many algorithms related to cloud computing have been proposed some of them have been surveyed in this paper. Chunlin Li et al. [1] proposed a load-balancing with resource optimization. In this model, the author has the attributes such as total cost, SLA, and user preferences to optimize the resource in an edge cloud environment. The migration strategy has been used for data movement with resource optimization and achieved load balancing in the age cloud environment.

The adaptive dragonfly algorithm (ADA) was proposed as a meta-heuristic task scheduling algorithm for the cloud by P. Neelima et al. [2]. This algorithm is the hybrid approach of the dragonfly and Firefly algorithm. To get the optimized result, the author has considered multiple parameters such as Processing Cost, Completion Time, and Load in the virtual machine.

Wenwei Cai et al. [3] proposed a massively cost-effective and load-balancing scheduling mechanism to maximize system throughput. The average waiting time and job response time are determined by modeling the entire cluster of computers as a queueing system and applying the principles of queuing theory to the problem. The theory of convex optimization is then applied to the problem of assigning tasks to workers in order to create a system for distributing workloads fairly.

S. Peer Mohamed et al. [4] proposed another meta heuristic approach to address load balancing for cloud. Author proposed the load balancing algorithm for optimized and enhanced cloud infrastructure and showing the issues regarding the scheduling. this technique is illustrating static and dynamic condition based on the virtual memory parameters that will be fixed or chosen runtime. It deals with both static and dynamic properties of submitted tasks.

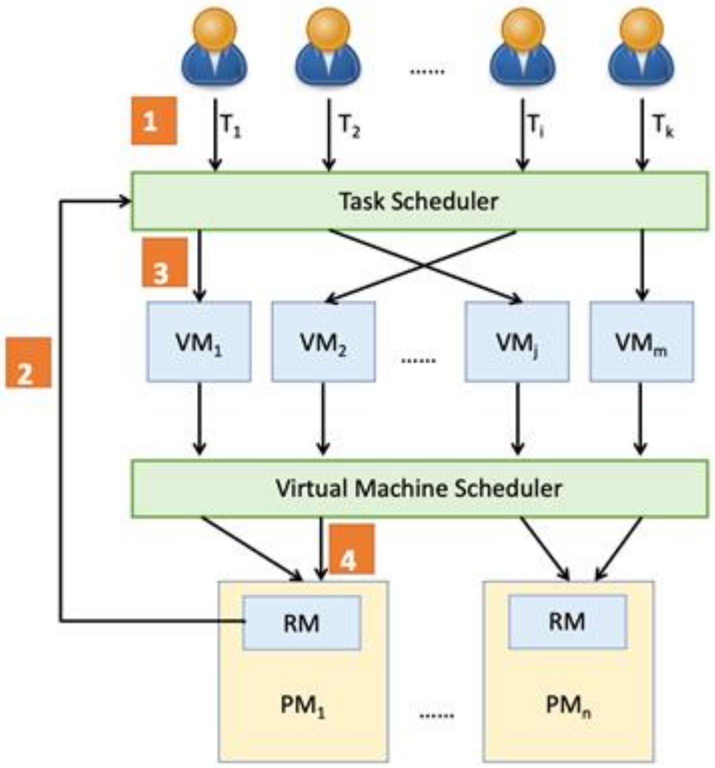

Fig 1: Framework of Cloud Computing with Physical Memory and Task Schedular

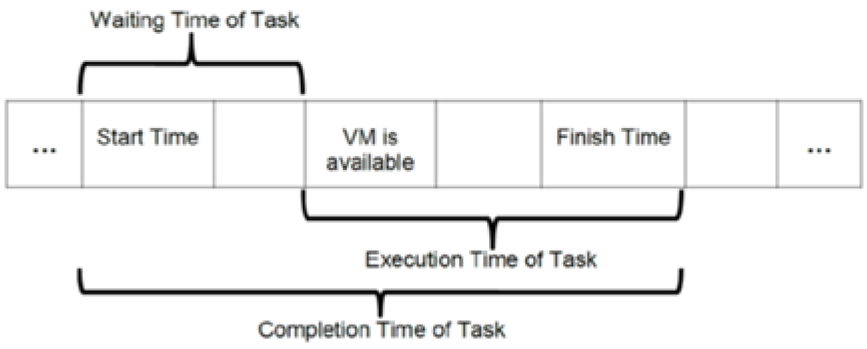

Fig 2: Time line of the Completion Time of the Task.

Mirza Mohamed et al. [5] proposed a technique that support real time applications to reduce the resource cost and delay for fog cloud environment. In this an optimised model For load balancing and three layer cooperative for computing environment has been proposed. in the proposed model link bandwidth and virtual machines processing capacity has been considered. The load balancing algorithm proposed by Muhammad Junaid et al. [6] is a modified version of Cat Swarm Optimization (MCSO) that integrates SVM and popularly known as Data File Type Formatting (DFTF). this algorithm is algorithm. The SVM classifier differentiate between various sources such as audio video images and text. The task is divided up among the virtual machines using MCSO. The enhanced efficient strategy for the cloud computing environment was presented by Rajagopal et al. [7] to distribute the dynamic load over multiple Virtual Machines (VMs). It allocates the task to the virtual machine Based on the makes pain time off each virtual machine and it's assigned task. Author also claim that the improved efficient scheme (IES) gives better completion time then other schemes.

Shanchen Pang et al. [8] proposed a hybrid load balancing technique that combines Estimation of Distribution Algorithm (EDA) and GA to enhance cloud computing's load balancing capabilities and speed up work completion. Assigning tasks to virtual machines is best accomplished by first using the probability and sampling approach of EDA to develop a workable solution, and then expanding the search space through mutation and crossover operation in a genetic algorithm. Authors also claim that it has fast coverage speed and efficient search capabilities. Altaf Hussain et al. [9] presented the load Balancing scheduler based on SLA for the heterogeneous cloud computing architecture to decrease the execution time and exaction cost of each submitted work. This method is cost-effective and helps cloud users meet their SLAs. It took into account the task's execution cost while deciding which virtual machine to assign it to. Amrita Jyoti et al. [10] present a new method for dynamic load balancing in the cloud computing environment to improve QoS performance in areas such as energy, time delay, reaction time, and so on. For dynamic resource allocation, we used a multi-agent best deep reinforce meant learning approach. In this situation, a virtual machine is selected to perform the task based on its relative importance. A second dynamic approach is described for scheduling the task days, and it is based on the dynamic optimal load aware service broker and the global user agent.

Lingfu Kong et al. [11] proposed Heuristic algorithm for heterogeneous virtual machine based cloud environment to provide low make-span and monetary cost with greater resource utilization. In the proposed approach optimal completion time and earlier finishing time attributes are used for virtual machine to achieve start scheduling and load balancing efficiently. The catastrophic genetic algorithm (CGA) was introduced by Shudong Wang et al. [12] as a means of achieving a global optimum solution in the work scheduling algorithm for edge devices and cloud computing infrastructureThis algorithm's attributes are the time it takes to finish a task and the penalty it incurs for failing to finish a task on time. The algorithm's performance was enhanced by employing an optimized mutation and crossover procedure and a roulette selection approach. Li Liu et al. [13] presented Multi objective optimization task scheduling algorithm where unconstraint hand time deadline constraint scenarios has been considered. The attribute for that are scheduling algorithm is heterogeneous earliest finish time for order preferences and idle solution method. Vijayakumar et al. [14] proposed another meta-heuristic approach they name the Dragonfly Optimization algorithm. In this method, a constraint measure is used to accomplish load balancing. The workload and available resources of a virtual machine are compared, and if the workload exceeds the threshold value, the jobs are transferred to a machine with a lower workload. Capacity and load in each virtual machine are used to balance the load of the virtual machines.

Jean Pepe et al. [15] proposed low cost and low time complexity load balancing algorithm based on PSO. In this algorithm based on the completion time virtual memory is assigned to the task in hydrogenous virtual memory environment. the particle positioning is updated based on the load balancing scheme. Kaushik et al. [17] also proposed a meta-heuristic algorithm To optimize the load balancing in cloud server with multiple virtual machines and tasks. This algorithm is based on dominant Firefly algorithm to enhance the accuracy and to balance the load. In a mobile and cloud computing setting, the authors believe their proposed technique can increase throughput and decrease response time. Keng-Mao et al. [18] offer a hybrid meta heuristic technique to managing the variable workload in the cloud computing environment by combining Ant Colony optimization (ACO) and PSO. They will move the task to the virtual machine during runtime after considering the task's historical performance data. To be specific, Fahimeh Ramezani et al. [19]. It was suggested that the meta-heuristic technique of PSO be used for task-based load balancing. This approach is designed to reduce the burden of virtual machine migration. By shifting the additional workload to a virtual machine that is less overloaded, it is possible to accomplish the reduced migration overload.

It achieves the reduced migration overload by distributing the additional work across the virtual machines and putting the burden on those with less requests. Some sort of optimization method is employed in Firefly to achieve the best possible outcome and load distribution. the time to complete the task, the task in queue and completion cost are the attribute used to find the optimal result.

3. Problem Formulation

The typical cloud computing infrastructure framework is illustrated in figure 1. Here we assume that the cloud computing infrastructure consist of physical machines (PM_S ….), which is loaded with resource manager (RM), who's responsibility is to monitor the utilization of resources off physical machine and gather the statistics of every physical machine such as availability of physical machine utilization of resources and the status of virtual machine. each physical machine can have many virtual machines which is taken care by virtual machine monitor (VM_m). The virtual machine employs the virtual machine scheduler to map the Physical Machine (PM) using the Resource Manager (RM). In the diagram a sequence of actions has been shown in the Figure 1. (1) M number of tasks are available (such as T_1 T_2 $T_3 \ldots T_M$) for the scheduling. Task scheduler gets the M number of users task initially. (2) Resource manager gives the feedback to the task scheduler For the status Of the physical machine hand it's remaining resources. (3) The task scheduler initiate the scheduling of the user task based on the feedback received by the resource manager and the user request received by the service provider. (4) The Virtual Machine Scheduler (VM Scheduler) allocates the VMs on the selected PM. In the proposed task scheduling algorithm, initially it calculates the completion time for executing the task. The completion time is calculated using execution time as well as the waiting time of the user task. The waiting time of the user task is the time duration when the task is not allotted the CPU. Feedback from the RM on the capacity of the VM used to perform the tasks and the number of instructions in each task can be used to estimate the total execution time. The task has been executed has been executed into the virtual machine if the executing task's completion time is within the deadline define by the user. After the execution of the user task, the execution cost of the task in each existing VM has been calculated by the task scheduler. By allocating the most appropriate virtual machine to run the user's work, the task scheduler can reduce the overall cost of execution. At last, the user will get the result of the executed tasks by the task scheduler.

The following assumptions used in the proposed algorithm:

- Each user task In the virtual machine is completed without any interrupt after initialization of execution i.e. we apply non-preemptive scheduling.
- The execution time duration of each users task is dependent on virtual machine's capacity.
- All users task are independent to each other i.e. full isolation is applied.
- Based on the assigned virtual machine which fulfil the requirements of the user task is allowed to process.
- More than one user's tasks are allowed in each virtual machines.

Inputs to the Algorithm:

- The different set of Tasks $T = \{T_1, ..., T_i, ..., T_M\}$ can be initiate and defined, here in the task set $i \in [1, M]$ and the total number of tasks represnetd as M. Every user's task can be represented as following tuples, for any task i it presnetd as T_i <$inst_i$, $deadline_i$, arv_i>, where $inst_i$ is the task size which can be calculated by million of instructions ($Million_i$), $deadline_i$ is the deadline of the task and arv_i is the task arrival time.

- The different set of Vitural Machines $VM = \{VM_1,...,VM_j,...,VM_V\}$ can be defined and initialted with $j \in [1,V]$ here, the total number of Vitrual Machines are V . Every Virtual Machine VM_j can be reperented by the tuples is described as VM_j <C_j,$Price_j$>. The values of C_j here stand in for the VM's storage capacity. The capacity, expressed in "million instructions per second" (MIPS), is merely the processing speed. The $Price_j$ per hour could be used to represent the time spent on the virtual machine's task.

Output of the Algorithm:

- With the criteria fulfilled by the algorithm the choosen Virtual Machine of each task.

Objective Function of the Alorithm:

- The purpose of the FSOTS algorithm is to assign the appropriate virtual machine to the user stars so that the execution cost (*EC_{ij}*)could be minimised. The objective function is shown in the equation below:

$$\min f = \sum_{i=1}^{M} EC_{ij} \quad (1)$$

Execution Cost:

- The time taken by the VM to carry out the operation is known as its execution *cost (EC_{ij})*. This is determined by the task's *CT_{ij}* and the cost of the virtual machine *VM_j* and represented by the notation *$Price_j$*. This can be represented using the equation below:

$$EC_{ij} = Price_j * CT_{ij} \quad (2)$$

Completion Time:

- The arrival time of the task *i* and the execution time of the task in virtual machine *VM_j* can be used to determine the completion time (*CT_{ij})* of the task that is executed in the virtual machine. As shown in Figure 2, the task completion time can be determined using the following equation:

$$CT_{ij} = arv_i + ET_{ij} \quad (3)$$

In the above equation the arrival time of the user's take *i* in to the Vitural Machine is reprented as *arv_i* and execution time of any user's task *i* is *ET_{ij}*.

The execution time is the amount of time taken by the VM to finish task execution and can be expressed as follows:

$$ET_{ij} = \frac{inst_i}{C_j} \quad (4)$$

In the avove equation the number of instructions consists of that task i that need to be exected in the virual machine *VM_j* is *$inst_i$* and the capacity of Virtual Machine *VM_j* can be denoted as *C_j*. The capacity of the VM can be calculated using the formula shown in the quation below:

$$C_j = (Pe_j * mips_j) \quad (5)$$

In the above equation the number or core of processor available in the VM can be denoted as *Pe_j* and the millions of instruction per second value by those proceesor can be denoted as *$mips_j$*.

I. Basic Firefly Swarm Optimization Algorithm

Firefly swarm optimization (FSO) is an optimization algorithm that is based on the intelligence and population of fireflies. It is a distributed approach to find the optimal solution and more scalable than other metaheuristic algorithms. FSO algorithm can be applied in dynamic VM allocation and load balancing in cloud computing environment. This algorithm is completely based on the behaviour of the fireflies and how they react with different phenomena. The fireflies move towards another Firefly if they find more illumination then others. It is also based on the distance and the intensity of elimination. as in the normal phenomena Firefly attract with other fireflies based on how much light they're detecting. in the Firefly some optimization algorithm the basic objective function is based on the illumination reduce by the fireflies it means the higher the illumination the higher the chance to select that Firefly as neighbor. It is a probabilistic algorithm and find the estimated cost based on this objective function. we can say that instead of global Maxima it is more towards the local Maxima. hence we can find many clusters of fireflies which creates the sub group based on the objective function. The FSO algorithm initially positioned the fireflies randomly with random elimination value this elimination also depends on the distance between the fireflies. FSO algorithm has four phases such as initialization phase, illumination updation Phase and Displacement Phase and finally Local Radial Updation Phase. The detailed Firefly Swarm Optimization algorithm is illustrated in Algorithm 1.

Algorithm 1: Firefly Swarm Optimization

```
Initialise parameters $\beta, p, s, z_t$
catchcatch:end $\forall j, set l_j(0) = l_0$
$\forall j, set \gamma_d^j(0) = \gamma_0$
while termination condition not met do
    for $j \in m$ do
        $l_j(t+1) = (1-p) * l_j(t) + \gamma f(x_j(t+1))$
        $N_j(t) = \{n : ||x_n(t) - x_j(t)|| \leq \gamma_d^j(t); l_j(t) \geq l_n(t)\}$
        for $n \in N_j(t)$ do
            $p_{jn}(t) = \frac{l_n(t) - l_j(t)}{\sum_{k \in N_j(t)} \{l_k(t) - l_j(t)\}}$
        end
        $x_j(t+1) = x_j(t) + s\left(\frac{x_n(t) - x_j(t)}{||x_n(t) - x_j(t)||}\right)$

        $\gamma_d^j(t+1) = min\{\gamma_s, max\{0, \gamma_d^j(t) + \beta(n_t - |N_j(t)|)\}\}$
    end
    $t = t + 1$
end
return Optimal Solution
```

4. Proposed Modified Firefly Swarm Optimization algorithm

Firefly Swarm Optimization algorithm finds the virtual machine for each task to minimize the execution cost. In the algorithm each virtual machine can be visualised as the fireflies and the virtual machine execution cost represents by the illumination. Based on the basic fireflies behaviour, they always attract towards the other fireflies having the more illumination. In the modified proposed Firefly some optimization algorithm unlike basic Firefly some optimization The objective function is based on the lower execution cost which is opposite to the characteristics of maximum illumination. The detailed discussion on proposed algorithm is given below:

A. Initialization Phase

In this phase Initial population has been considered which are the number of fireflies considered as the potential solutions. this initial population which consist of the fireflies are randomly generated. As in the algorithm Firefly signifies virtual machines, so each virtual machine (*VM_j*) Consist of its location at any time *t*, *$x_j(t)$*. This location signifies the completion time of the task in any virtual machine (*VM_j*). The completion time is nothing but the time taken by the virtual machine to execute any task t. The illumination value of any Firefly at time t is represented as *$l_j(t)$*. The illumination is equal to the execution cost of the virtual machine. The other parameters used in the algorithm are Maximum sensor range represented as γ_s in the size of displacement step s, the local radial range at any time *t* can be represented as $\gamma^j_d(t)$, the desired number of neighbors are n_t. In the algorithm, the illumination decade coefficient *p* with the range ($0 < p < 1$) can be used, the illumination enhancement coefficient can be represented as γ and the neighborhood rate of change would be presented as *β*.

Algorithm 2: Firefly Swarm Optimization based Task Scheduling

```
catchcatch:end Input : $vmList, taskList$
Output: $selectedVm$
Initialise parameters $\beta, p, s, n_t, \gamma_d^j$
$t = 1$
for $i \in taskList$ do
    $taski \to vm_j$ randomly
end
for $j \in vmList$ do
    $l_j(t) = Price_j * CT_{ij}$
end
for $i \in taskList$ do
    while termination condition not met do
        for $j \in vmList$ do
            $l_j(t+1) = updateLuc(j) by Eq.(6)$
            $N_j(t) =$
              $getVmNeighbours(j, n_t) by Eq.(7)$
            for $n \in N_j(t)$ do
                $p_{jn}(t) = calcProb() by Eq.(8)$
            end
            Select the highest $p_{jn}(t)$
            selectedVm $\leftarrow$ highest $p_{jn}(t)$
            Update $\gamma_d^j by Eq.(10)$
        end
        $t = t + 1$
    end
    if selectedVm $\neq$ NULL then
        return selectedVm
end
```

Algorithm 3: $getVmNeighbours(j, n_t)$

```
$N_j \to NULL$
$l_j(t) = Price_j * CT_{ij}$
for $n \in n_t$ do
    while $n \leq n_t$ do
        $l_n(t) = Price_n * CT_{in}$
        if $l_n(t) < l_j(t)$ then
            if $||CT_{in}(t) - CT_{ij}(t)|| \leq deadline_i$
              then
                $N_j \leftarrow n$
    end
end
```

B. Illumination updation Phase

At the initial stage, the objective function value $f(x_j(t+1))$ of each virtual machine VM_j could be converted into the illumination value $l_j(t+1)$ With reference to the equation shown below.

$$l_j(t+1) = (1-p) * l_j(t) + \gamma f(x_j(t+1)) \quad (6)$$

In the above equation, $l_j(t)$ is the illumination value of the concern virtual machine (VM_j)at any time t. Also in the equation the objective function at time $t + 1$, represents $f(x_j(t+1))$. Hence the elimination value of the virtual machines are updated according to the value given by the objective function. In the algorithm the lower illumination value represents the lower execution cost which is the goal of the algorithm to minimize. the illumination value minimized based on the time to illustrate the decay in the basic algorithm.

C. Displacement Phase

In this phase, The Firefly decide to move towards one of its neighbor Firefly to illustrate the displacement of virtual machine with the task. the displacement between virtual machine VM_j and the task t, are done with the objective function off minimum execution cost represents the lower illumination value. The displacement is also based on task deadline represented by Local radial range γ_d.

The displacement phase further divided into four steps:

Step 1. Neighbors Finding:

In this step appropriate neighbor could be found based on the criteria such as deadline for executing the task which can be represented as the equation below:

$$N_j(t) = \{n : ||x_n(t) - x_j(t)|| \leq \gamma_d^j(t); l_j(t) \geq l$$

In the above equation an is the ID of the neighbor of the virtual machine, $l_n(t)$ & $l_j(t)$ are the illumination values for Virtual machine n (VM_n) and Virtual machine j (VM_j) and $x_n(t)$ & $x_j(t)$ are completion time of Virtual machine n (VM_n) and Virtual machine j (VM_j).

Step 2. Probability Calculation:

In this step probability of each Virtual Machine (VM_j), to displaced towards the neighbor with the lowest illumination value can be calculated using the equation below:

$$p_{jn}(t) = \frac{l_n(t) - l_j(t)}{\sum_{k \in N_j(t)} \{l_k(t) - l_j(t)\}} \quad (8)$$

In the above question the probability of the task to displace from virtual machine j (VM_j) to virtual machine n (VM_n) can be represented as $p_{jn}(t)$.

Step 3. Neighbor Selection:

In this step, The users task which is located in virtual machine J (VM_j) selects a virtual machine (VM_n) as neighbor set using the highest probability than the other neighbor virtual machine. After this phase, each task has assigned it's optimum virtual machine, which is a significant step to achieve load balancing in the cloud computing environment having physical machines.

Step 4. Displacement:

In this step related to each VM the completion time of each allocated task is calculated using the equation below:

$$x_j(t+1) = x_j(t) + s\left(\frac{x_n(t) - x_j(t)}{||x_n(t) - x_j(t)||}\right) \quad (9)$$

In the above equation, the current and the new completion time of the VM_j can be represented as $x_j(t)$ & $x_j(t + 1)$. For future selection of neighbor virtual machine the importance of calculating the new completion time is more.

D. Local Radial Updation Phase

In this phase, the deadline of each task which is represented as the local radial γ^j_d need to be updated for computing the future task scheduling to the virtual machine. For updating the local radial the equation is shown below:

$$\gamma_d^j(t+1) = min\{\gamma_s, max\{0, \gamma_d^j(t) + \beta(n_t - |N_j(t)|)\}\} \quad (10)$$

The Algorithm 2, illustrates the proposed task scheduling algorithm. In the algorithm the input parameter are used as the details of taskList and vmList. In line 1 parameter are initialised. A virtual machine is randomly initialized when the user's task need to process, as shown in line 3 – 4 in the algorithm. Based on the execution cost by calculating the illumination value in our case virtual machine gets the new position as in the line 5 – 6. In line 7 each task T_i, from the taskList has to schedule in the virtual machine. the elimination value at time t, $l_j(t)$, which is the execution cost of any virtual machine j need to be updated as shown in line 10. In line 11 the neighbor set of any virtual machine j, $N_j(t)$ will be calculated based on the lower execution cost then the current virtual machine and also that fulfil the condition of deadline. The user predefined the size of the neighbours set at any time t, n_t. The illumination of each firefly in the form of execution cost $l_n(t)$ using the cost and the completion time will be calculated as depicted in line 2 of Algorithm 3. To adding the virtual machine (VM_n) into the neighbor list the execution cost of the virtual machine along with the deadline of the task is used as written in line 3 – 8 in algorithm 3. As soon in line 12 – 15, the selection of virtual machine would be done based on the highest priority from the neighbor list. At last local radio γ_d would be updated from the neighbours list as in line 16. As depicted in line 18 – 19 the algorithm is terminated if no significant improvement will be done in the execution cost.

5. Simulation Results & Discussion

The proposed technique is tested using the CloudSim simulator. This is the widely used simulator for cloud computing architecture provides the representation of data centres and virtual machine. It also supports task scheduling policies, the virtual machine placement algorithm, selection and mapping of the task with the virtual machine, power models and provides different scheduling scenarios to simulate real time situations. The task scheduling algorithm is deployed in the clouds and by including the virtual machine detail and the task scheduling criteria. With two type of physical machines a data centre is created inside the simulator. Table 1 & Table 2 shows the attributes of data centre and physical machines. The parameter used in the proposed modified Firefly Swarm Optimization base task Scheduling algorithm are standard and taken from the existing papers as shown in Table 3. The simulation experiments of the proposed FSOTS along with three existing task scheduling algorithms using the meta-heuristic algorithm such as PSO[15], DFO [2] and BFA [4]. The performance of proposed FSOTS was compared with the PSO, DFO and BFA in terms of different matrices such as total Completion Time, Execution Cost and Resource Utilization.

TABLE I. CONFIGURATION OF THE DATA CENTRE

Parameter	**RAM**	**Storage**	**Bandwidth**	**VM Scheduler**	**VMM**
Value	2 GB	1 TB	10 GB	Time Sharing	Xen

TABLE II. CONFIGURATION OF THE PHYSICAL MACHINE

Physical Machine	**Processor**	**Core**	**MIPS**
PM1	Intel Core 2 Extreme X6800	2	27079
PM2	Intel Core i7 Extreme 3960X	6	177730

TABLE III. INITIAL VALUE OF THE PARAMETERS FOR THE PROPOSED (FSOTS) ALGORITHM

Parameter	p	R	β	n_t	s	l_0
Value	0.4	0.6	0.08	5	0.03	0.05

A. Total Completion Time

In a VM, the total completion time of a task is determined based on the submission time, execution time, and waiting time. The static and dynamic workload into the cloud computing infrastructure has been considered. Static workload is when the task is submitted beforehand to the mapping with the virtual machine, whereas dynamic workload is when the task is submitted at runtime. Apart from that in the simulation mainly two scenario has been considered in the first scenario number of tasks are varying whereas VMs are fixed. In the second scenario, the number of assignments is fixed while the number of VMs varies.

Scenario 1. Fixed Virtual Machine & Varying User's Tasks:

Five virtual machines are used regardless of the user's workload growth from 100 to 600. Each virtual machine's overall task completion time has been measured in milliseconds. Figures 3 and 4 show the number of tasks on the X-axis and the total time to complete the tasks on the Y-axis in milliseconds. One case with a static workload is explored in Figure 3, and it is found that the suggested (FSOTS) algorithm quickly maps the task with a virtual machine in a short amount of time. As can be shown in Figure 4, the suggested (FSOTS) method not only performs better than other current algorithms in the cloud computing environment, but also handles the dynamic summation of tasks. It shows that with both the workload condition the proposed algorithm can give better and optimal result.

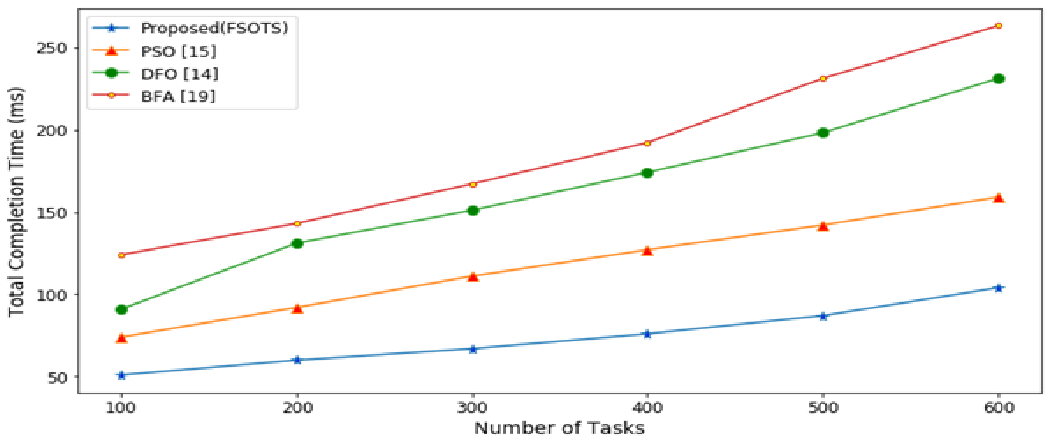

Fig 3. Total Completion Time (Static Workload): With Fixed Virtual Machines & Varying Tasks

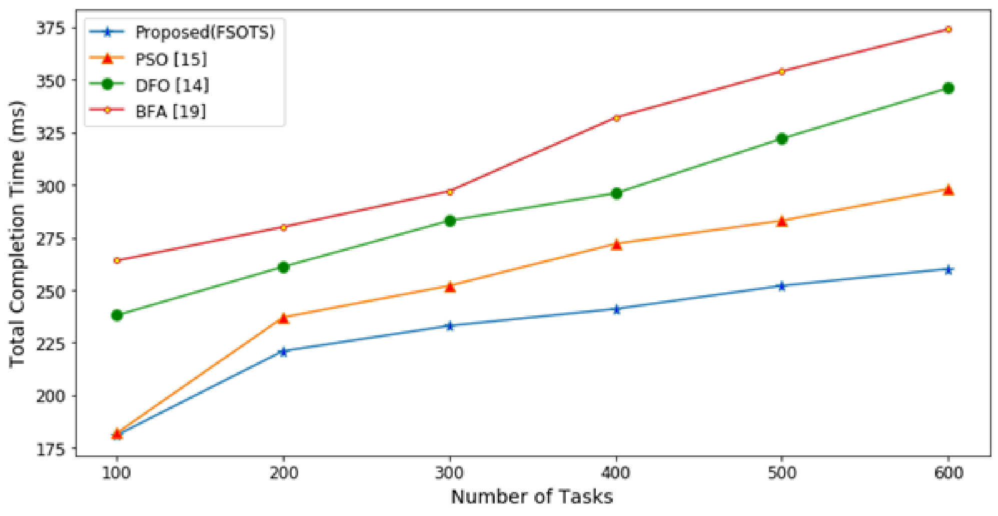

Fig 4. Total Completion Time (Dynamic Workload): With Fixed Virtual Machines & Varying Tasks

Scenario 2. Fixed User's Tasks & varying Virtual Machines:

In this the user's tasks is fixed to 100 and virtual machine is linearly increasing from 5 virtual machines to 20 virtual machines. In Figure 5 the completion time in milliseconds has been calculated with variable number of virtual machines. here X-axis denotes the Number of Virtual Machines (VMs) and Y-axis denotes the Total Completion Time (ms). This experiment is conducted with static workload of the tasks and found that the proposed algorithm (FSOTS) outperformed the existing protocol in terms of total completion time. We have also observed that in comparison of less number of virtual machines when we linearly increase the virtual machine the completion time is almost constant. it indicates that if the task is limited the completion time doesn't vary for greater number of virtual machines.

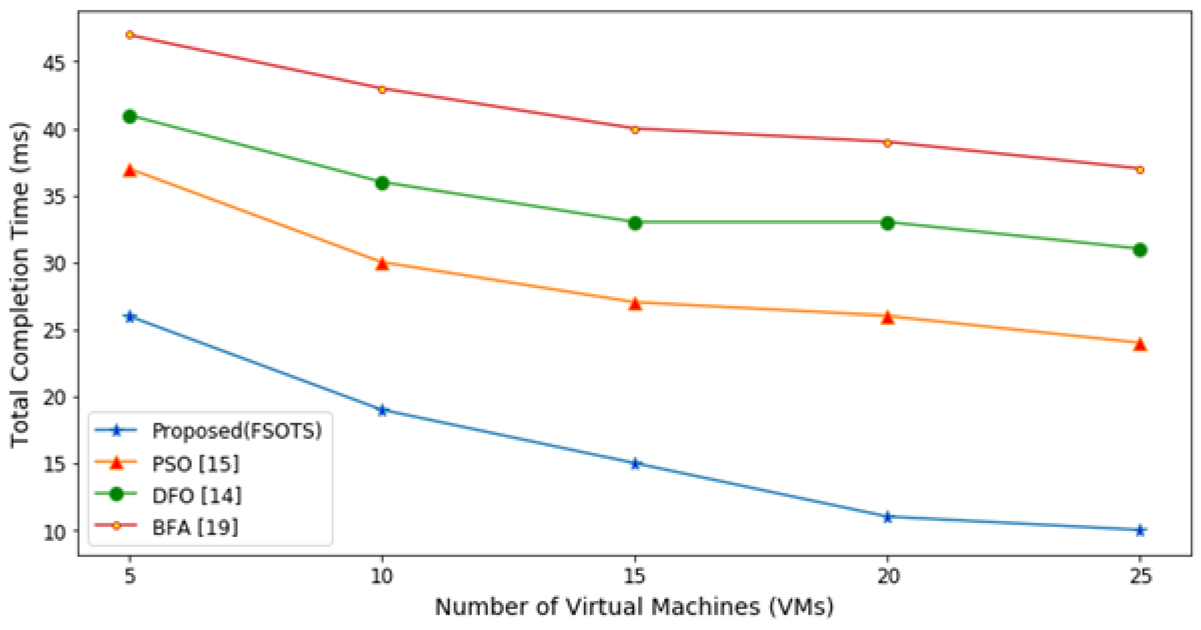

Fig 5. Total Completion Time (Static Workload): With Fixed User's Tasks & Varying Virtual Machines

B. Execution Cost

The time required to perform the work in the VM and the type of virtual machine can be used to determine the cost of execution. In this case, we have considered a fixed number of virtual machines of five, with the number of users' tasks varying and progressively increasing from 100 to 600. The workload into the cloud computing is fixed. In figure 6 the total execution cost with different number of user task is illustrated. As shown in the results, the suggested method (FSOTS) has a lower execution cost than the present approach, which gradually increases as the number of tasks increases.

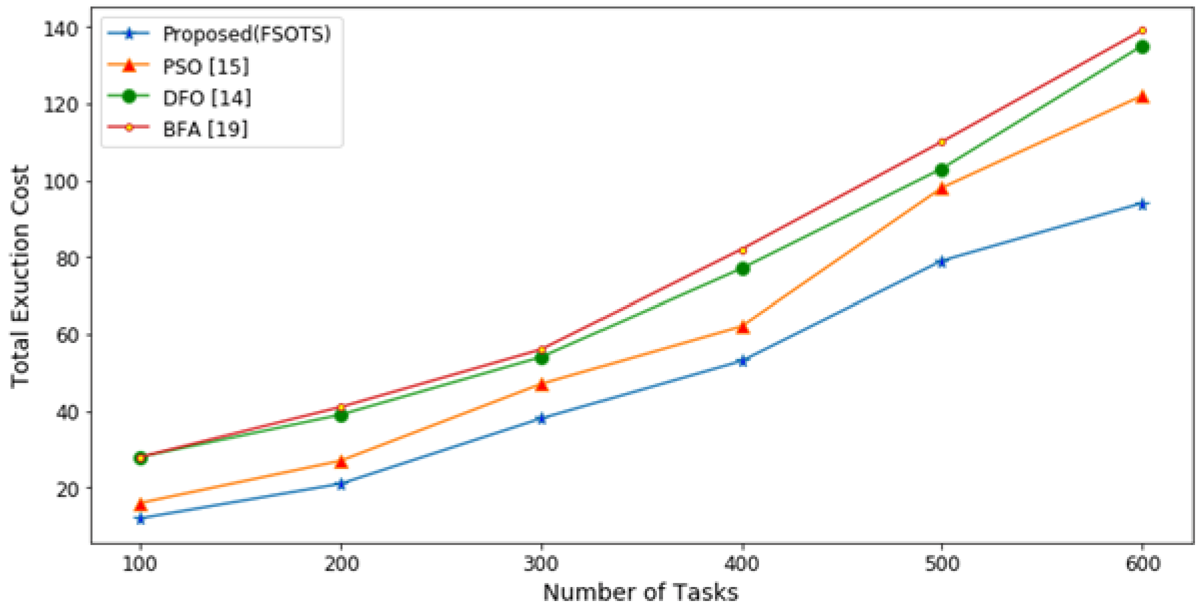

Fig 6. Total Execution Cost (Static Workload): With Fixed Virtual Machines & Varying User's Tasks

C. Resource Utilization

In the physical machine the resource given to the task for execution is virtual machine. we need to find out how the virtual machine could be utilised during the course of the task scheduling. The data center can calculate information about the load applied to the virtual machine. The load of the virtual machine can be determined using the standard deviation using the calculation below:

$$\sigma_j = \sqrt{\frac{\sum_{\mathrm{V}}^{\mathrm{j=0}} (\mathrm{CT_{ij}} - \mu_j)^2}{V}} \tag{11}$$

In the equation σ_j denotes the standard deviation of the virtual machine load, number of virtual machines can be denoted as V. The time of completion of any task execution Is denoted as $\mathrm{CT_{ij}}$. which is already calculated in the Equation 3. the mean completion time of virtual machine we is denoted as μ_j that is calculated with the equation below:

$$\mu_j = \frac{\sum_{j=1}^{V} CT_{ij}}{V} \tag{12}$$

As shown in the equation if the μ_j less than the σ_j, then the cloud computing system is in the imbalance state and if μ_j greater than or equal to the σ_j, then the cloud computing system is in the balance state. As shown in the Table 4 the proposed protocol (FSOTS) and PSO gives the result which shows that the cloud computing system is in the balanced state. Whereas DFO and BFA shows the result which takes the cloud computing system into imbalance state. it also denotes that in the proposed algorithm the resource utilisation is optimally achieved.

TABLE IV. STANDARD DEVIATION AND MEAN COMPLETION TIME

Algorithms	**μ**	**σ**
Proposed (FSOTS)	41.20	23.04
PSO	41.20	33.33
DFO	41.20	55.88
BFA	41.20	91.41

6. Conclusion

The meta-heuristic approach was employed for scheduling the tasks and load balancing among the VMs. The decision will be taken based on the Execution Cost and the Deadline submitted along with each user's task. The modified FSO algorithm is able to return the selected VM which can be mapped with the users task. The proposed method is implemented alongside the existing algorithm using the CloudSim simulator, and the results reveal the efficiency of proposed algorithm ascompared to existing protocol in terms of completion time, execution cost, and resource utilization. The resulting graph is constructed using several situations, such as a fixed user task and a varied virtual machine, as well as a varying user task and a fixed virtual machine with static and dynamic task submission into the cloud computing environment. The proposed approach will function in both static and dynamic task submission contexts. In future, further enhancement could be done and achieve more optimum result by using some hybrid meta-heuristic approach in a static and dynamic workload environment.

References

[1] Chunlin Li, Jianhang Tang and Youlong Luo, "Service Cost-based Resource Optimization and Load Balancing for Edge and Cloud Environment," Knowledge and Information Systems, Springer, vol. 62, 2020, pp. 4255-4275, doi:https://doi.org/10.1007/s10115-020-01489-6.

[2] P. Neelima and A. Rama Mohan Reddy, "An Efficient Load Balancing System using Adaptive Dragonfly Algorithm in cloud computing," Cluster Computing, Springer, vol. 23, 2020, pp. 2891-2899, doi:https://doi.org/10.1007/s10586-020-03054-w.

[3] Wenwei Cai, Jiaxian Zhu, Weihua Bai, Weiwei Lin, Naqin Zhou and Keqin Li, "A Cost saving and Load balancing Task Scheduling Model for Computational biology in Heterogeneous Cloud Datacenters," The Journal of Supercomputing, Springer, vol. 76, 2020, pp. 6113-6139, doi:https://doi.org/10.1007/s11227-020-03305-y.

[4] S. Peer Mohamed Ziyath and S. Senthilkumar, "MHO: Meta Heuristic Optimization Applied Task Scheduling with Load Balancing Technique for Cloud Infrastructure Services," Journal of Ambient Intelligence and Humanized Computing, Springer, July 2020, pp. 1868-5145, doi:https://doi.org/10.1007/s12652-020-02282-7.

[5] Mirza Mohamed Shahriar Maswood , Md. Rahinur Rahman, Abdullah G. Alharbi, and Deep Medhi, "A Novel Strategy to Achieve Bandwidth Cost Reduction and Load Balancing in a Cooperative Three-Layer Fog- Cloud Computing Environment," IEEE Access, vol. 8, 2020, pp. 113737-113750, doi:10.1109/ACCESS.2020.3003263.

[6] Muhammad Junaid, Adnan Sohail, Rao Naveed Bin Rais, Adeel Ahmed, Osman Khalid, Imran Ali Khan and Syed Sajid Hu, "Modeling an Optimized Approach for Load Balancing in Cloud," IEEE Access, vol. 8, 2020, pp. 173208-173226, doi:10.1109/ACCESS.2020.3024113.

[7] T. K. P. Rajagopal, M. Venkatesan and A. Rajivkannan, “An Improved Efficient Dynamic Load Balancing Scheme Under Heterogeneous Networks in Hybrid Cloud Environment,” Wireless Personal Communications, Springer, vol. 111, 2020, pp. 1837 - 1851, doi:https://doi.org/10.1007/s11277-019-06960-4.

[8] Shanchen Pang, Wenhao Li, Hua He, Zhiguang Shan and Xun Wang, “An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing,” IEEE Access, vol. 7, oct. 2019, pp. 146379 - 146389, doi: 10.1109/ACCESS.2019.2946216.

[9] Altaf Hussain, Muhammad Aleem, Muhammad Azhar Iqbal, Muhammad Arshad Islam, “SLA-RALBA: cost-efficient and resource-aware load balancing algorithm for cloud computing,” The Journal of Super- computing, Springer, vol. 75, June. 2019, pp. 6777 - 6803, doi: https://doi.org/10.1007/s11227-019-02916-4.

[10] Amrita Jyoti and Manish Shrimali, “Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing,” Cluster Computing, Springer, vol. 23, 2020, pp. 377 - 395, doi: https://doi.org/10.1007/s10586-019-02928-y.

[11] Lingfu Kong, Jean Pepe Buanga Mapetu and Zhen Chen, “Heuristic Load Balancing Based Zero Imbalance Mechanism in Cloud Computing,” Journal of Grid Computing, Springer, vol. 18, June 2019, pp. 123 - 148, doi:https://doi.org/10.1007/s10723-019-09486-y.

[12] Shudong Wang, Yanqing Li, Shanchen Pang, Qinghua Lu, Shuyu Wang and Jianli Zhao, “A Task Scheduling Strategy in Edge-Cloud Collaborative Scenario Based on Deadline,” Scientific Programming, Hindawi, vol. 2020, Mar. 2020, pp. 1-9, doi: https://doi.org/10.1155/2020/3967847.

[13] Li Liu, Qi Fan, Rajkumar Buyya, “A Deadline-Constrained Multi-Objective Task Scheduling Algorithm in Mobile Cloud Environments,” IEEE Access, vol. 6, Sept. 2018, pp. 52982 - 52996, doi: 10.1109/ACCESS.2018.2870915.

[14] Vijayakumar Polepally and K. Shahu Chatrapati, “Dragonfly optimization and constraint measure-based load balancing in cloud computing,” Cluster Computing, Springer, vol. 22, Jan. 2019, pp. 1099–1111, doi: https://doi.org/10.1007/s10586-017-1056-4.

[15] Jean Pepe Buanga Mapetu, Zhen Chen and Lingfu Kong, “Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing,” Applied Intelligence, Springer, vol. 49, Sept. 2019, pp. 3308 - 3330, doi: https://doi.org/10.1007/s10489-019-01448-x.

[16] Sweekriti M Shettyand Sudheer Shetty, “Analysis of Load Balancing Cloud Data Centers,” Journal of Ambient Intelligence and Humanized Computing, Springer, vol. 1 - 9, Jan. 2019, doi:https://doi.org/10.1007/s12652-018-1106-7.

[17] Kaushik Sekaran, Mohammad S. Khan, Rizwan Patan, Amir H. Gandomi, Parimala Venkata Krishna, Suresh Kallam, “Improving the Response Time of M-Learning and Cloud Computing Environments Using a Dominant Firefly Approach,” IEEE Access, vol. 7, 2019, pp. 30203 - 30212, doi:10.1109/ACCESS.2019.2896253.

[18] Keng-Mao Cho, Pang-Wei Tsai, Chun-Wei Tsai, Chu-Sing Yang, “A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing,” Neural Computing and Applications, Springer, vol. 26, Dec. 2014, pp. 1297 - 1309, doi:https://doi.org/10.1007/s00521-014-1804-9.

[19] Fahimeh Ramezani, Jie Lu, Farookh Khadeer Hussain, “Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization,” International Journal of Parallel Programming, Springer, vol. 42, Oct. 2013, pp. 739 - 754, doi:https://doi.org/10.1007/s10766-013-0275-4.

[20] Pawan Kumar and Rakesh Kumar, “Issues and Challenges of Load Balancing Techniques in Cloud Computing: A Survey,” ACM Computing Survey, vol. 51, Number 6, Feb. 2019, pp. 1 - 35, doi:https://doi.org/10.1145/3281010.

[21] Shahbaz Afzal and G. Kavitha, “Load balancing in cloud computing – A hierarchical taxonomical classification,” Journal of Cloud Computing: Advances, Systems and Applications, Springer, vol. 8, Issue 22, Jan. 2019, pp. 1 - 24 , doi:https://doi.org/10.1186/s13677-019-0146-7.

[22] Ravanappan, P. ., Ilanchezhian, P. ., Chandrasekaran, N. ., Prabu, S. ., & Saranya, N. N. . (2023). Secure Blockchain Transactions for Electronic Health Records based on an Improved Attribute-Based Signature Scheme (IASS). International Journal on Recent and Innovation Trends in Computing and Communication, 11(4s), 77–83. https://doi.org/10.17762/ijritcc.v11i4s.6309

[23] Ahammad, D. S. K. H. (2022). Microarray Cancer Classification with Stacked Classifier in Machine

Learning Integrated Grid L1-Regulated Feature Selection. Machine Learning Applications in Engineering Education and Management, 2(1), 01–10. Retrieved from http://yashikajournals.com/index.php/mlaeem/article/view/18

[24] Dhablia, D., & Timande, S. (n.d.). Ensuring Data Integrity and Security in Cloud Storage.

[25] Dhabalia, D. (2019). A Brief Study of Windopower Renewable Energy Sources its Importance, Reviews, Benefits and Drwabacks. Journal of Innovative Research and Practice, 1(1), 01–05.