Section A-Research paper
ISSN2063-5346

OPPOSITIONAL DOWNHILL SIMPLEX MULTI-OBJECTIVE FIREBUG SWARM RESOURCE OPTIMIZED TASK SCHEDULING IN CLOUD

Mrs.J.Radha, Assistant Professor, Department of Computer Applications, PSG College of Arts & Science, Coimbatore

Dr.V.Saravanan, Professor & Head, Department of Information Technology, Hindusthan college of Arts and Science, Coimbatore

ABSTRACT

Cloud computing is the emerging paradigm and permits user to access computing resources as services through the internet. In cloud computing, hundreds of online users submit many tasks anytime and anyplace to dynamic resources. The processing of a large number of arriving tasks in the cloud consumes more response time. Many algorithms have been introduced to schedule the tasks to the server in the cloud environment. But, optimal scheduling of arriving tasks plays a major problem in cloud environments for better utilization of resources by assigning certain tasks to the particular virtual machine. Therefore a novel technique called an Oppositional downhill simplex Multi-Objective Firebug Swarm Resource Optimized Task Scheduling (ODSFSMROTS) is introduced for efficient cloud service rendering with minimal makespan and resource consumption. The main objective of the ODSFSMROTS technique is to balance the workload on computer resources by distributing the incoming user tasks to available resources with higher efficiency. In the ODSFSMROTS technique, the user sends the number of incoming user-requested tasks to the cloud server. Then, the ODSFSMROTS technique uses an Oppositional downhill simplex Multi-Objective Firebug Swarm Optimization algorithm to determine the resource optimal virtual machines for every input user requested tasks with minimal time consumption. For each virtual machine, the fitness function is evaluated with help of the downhill simplex method based on multi-objective such as energy, makespan, bandwidth, memory, and CPU. After that, the current best virtual machine is selected by using a steady-state selection approach based on fitness. Followed by, different behaviors of the Firebug Swarm are estimated based on the fitness function. Finally, the user task is assigned to the optimal virtual machine. In this way, efficient task scheduling is achieved with higher efficiency. Experimental

evaluation is carried out on factors such as task scheduling efficiency, makespan, throughput, and memory consumption with respect to a number of requested tasks.

Keywords: Cloud, task scheduling, Oppositional Firebug Swarm Multi-Objective Resource Optimization, downhill simplex method, steady state selection approach

1. INTRODUCTION

Cloud computing is the modern computing standard, which provides information technology (IT) services to the end users over the internet. In the cloud, a user task is mapped with an available resource to achieve a better result. Task scheduling is the process that is used to assign incoming tasks on virtual machines (VMs) of a cloud server based on its capacity of workload. Tasks are assigned to the server in such a way as to decrease the time delay of execution as well as response time. Similarly, there are several problems to be considered, such as optimization of energy resource utilization. Therefore, the task scheduling for the virtual machine process is the most important in managing the resources of a cloud data center. Many Task scheduling techniques have been done to allocate resources to the cloud platforms. A Task Schedule using Multi-Objective Grey Wolf Optimizer (TSMGWO) was developed in [1] to determine near-optimal task scheduling solutions. The designed technique increases the throughput and minimizes the makespan but it failed to focus on enhancing the TSMGWO approaches by using other parameters of the cloud environment like memory usage. Multi-objective task scheduling optimization was developed in [2] based on the Artificial Bee Colony Algorithm (ABC) with a Q-learning (MOABCQ) method. The designed algorithm reduces the makespan and increases the throughput as well as average resource utilization. But the performance of task sidling efficiency was not analyzed.

An improved whale optimization algorithm (IWC) was developed in [3] for increasing the scheduling efficiency and minimizing the task scheduling time. But it failed to develop an efficient scheduling system suitable for various task workloads in a cloud computing environment. A Task Scheduling-Decision Tree (TS-DT) algorithm was developed in [4] to execute the numerous applications' tasks. Though the algorithm reduces the makespan, performance parameters such as power consumption and scalability were not considered. An energy-aware algorithm for workflow scheduling algorithm was developed in [5] to address

multi-objectives such as energy consumption, resource utilization, and virtual machine migrations. However, it failed to improve its performance by saving more energy which enables the algorithm to obtain intelligent decisions in selecting virtual machines for task mapping. A Discrete variation of the Distributed Grey Wolf Optimizer (DGWO) was designed in [6] for assigning the dependent tasks to a virtual machine with the best makespan. But the performance of throughput was not improved.

A dynamic round-robin heuristic algorithm (DRRHA) was designed in [7] by using a round-robin algorithm for scheduling the tasks. However, the algorithm failed to achieve better performance, for holding a high number of tasks. A fuzzy self-defense algorithm was designed in [8] based on multi-objective task scheduling optimization to minimize the performance of completion time. But the efficiency of the task scheduling was not improved. A strong agile response task scheduling optimization algorithm was introduced in [9] for minimizing the time span of task scheduling. But it failed to analyze the energy consumption optimization for minimizing the time complexity and space complexity. A Whale Optimization algorithm was designed in [10] to schedule the tasks on to an appropriate virtual machine. But the designed algorithm failed to achieve higher throughput.

1.1 Our contribution

In order to solve the existing issues, a novel ODSFSMROTS is developed with the following contributions,

- In order to improve the task scheduling in the cloud for rendering the services, A novel technique ODSFSMROTS is introduced for minimizing the makespan and ensuring higher task scheduling efficiency.
- The ODSFSMROTS technique uses the Multi-Objective Firebug Swarm Resource Optimization in the novelty of Opposition-based learning, downhill simplex method, and steady-state selection approach.
- To find resource optimal virtual machines, Opposition-based learning is used to generate the opposite population in the search space for selecting the current best virtual machine.

- The downhill simplex method is applied to Multi-Objective Firebug Swarm Resource Optimization for measuring the fitness by finding the minimum and maximum resources availability of virtual machines.
- Depending on the fitness function, the steady-state selection approach is used for finding the current best virtual machine to minimize the time consumption of precise and accurate task scheduling. Therefore, the makespan is said to be reduced and achieve higher throughput.
- Finally, extensive and relative experiments are conducted to evaluate the quantitative analysis based on the different performance metrics.

1.2 Organization of paper:

The rest of this article is structured as follows: Section 2 discusses the literature review. In section 3, a description of the proposed ODSFSMROTS technique is presented with a neat diagram. In section 4, experimental settings and dataset description is presented. Section 4 provides the performance assessment of the proposed algorithm in comparison with existing scheduling techniques. And finally, section 6 presents the conclusions of the work.

2. RELATED WORKS

A metaheuristic approach for Dynamic Virtual Machine Allocation (MDVMA) was developed in [11] for optimized task scheduling with minimal energy usage and makespan. But it failed to focus on improving optimal scheduling for virtual machine selection in the cloud. An energy-efficient dynamic scheduling scheme (EDS) was designed in [12] for decreasing mean response time and reducing energy consumption. But the designed task scheduling failed to obtain better performance. A hybrid metaheuristic approach was developed in [13] for task scheduling to reduce the average completion time. But the scheduling process failed to process a maximum number of tasks.

A hybrid swarm optimization (HSO) algorithm was designed in [14] that integrates a combination of PSO and salp swarm optimization (SSO) to schedule the task for minimizing the execution time. But it failed to extend with QoS parameters to perform task scheduling in the cloud. A knowledge-driven multi-objective evolutionary algorithm was designed in [15] to

allocate the multiple tasks by balancing makespan and cost. But, it failed to achieve higher throughput while handling multiple tasks.

A Self-Adapting Task Scheduling Algorithm (ADATSA) was designed in [16]. But the efficiency of task scheduling was not improved. Adaptive Scheduling Algorithm-based Task Loading (ASA-TL) was developed in [17] for achieving higher performance with lesser response time. But it failed to implement the proposed scheme in the cloud server by means of virtual software resources such as energy, memory, and so on.

Elastic Scheduling for Microservices (ESMS) was developed in [18] that combines task scheduling with auto-scaling to reduce the cost of virtual machines. However, it failed to accurately predict the response time of each micro-service with different configurations.

A task scheduling-based queue algorithm (TSBQ) was introduced in [19] which consider the data transmission delay and reasonable task allocation policy. However, the performance of the makespan analysis remained unsolved. An enhanced sunflower optimization (ESFO) algorithm was designed in [20] for task scheduling and reducing the makespan as well as energy consumption. But, multi-objective optimization was not introduced for task scheduling in the cloud.

3. PROPOSAL METHODOLOGY

Cloud computing is an on-demand availability of system resources without direct management by the user. A cloud service provider delivers various on-demand services to the end users over the internet. Then the users access the cloud services at remote locations. In cloud computing, the task scheduling approaches directly affect the resource utilization efficiency of the fundamental system. Task scheduling is a significant process to enhance the overall performance of cloud computing. The service provider has to provide numerous users applications at different times. Different evolutionary algorithms have been designed to solve the task scheduling in the cloud. But, the makespan and resource utilization performance was not improved during task scheduling by using conventional population-based algorithms. Based on this motivation, a novel technique ODSFSMROTS is introduced.

Contrary to the existing optimization, the proposed ODSFSMROTS technique uses the opposition learning method to improve the performance of optimization techniques in low dimensions and also improve the scalability. The scalability is the measure of the capacity of the algorithm while considering the many tasks arrived at the cloud center. The proposed ODSFSMROTS technique solves the multiple objective functions as well as also uses the opposition-based learning concept to improve the task scheduling efficiency. The architecture of the ODSFSMROTS technique is shown in figure 1.

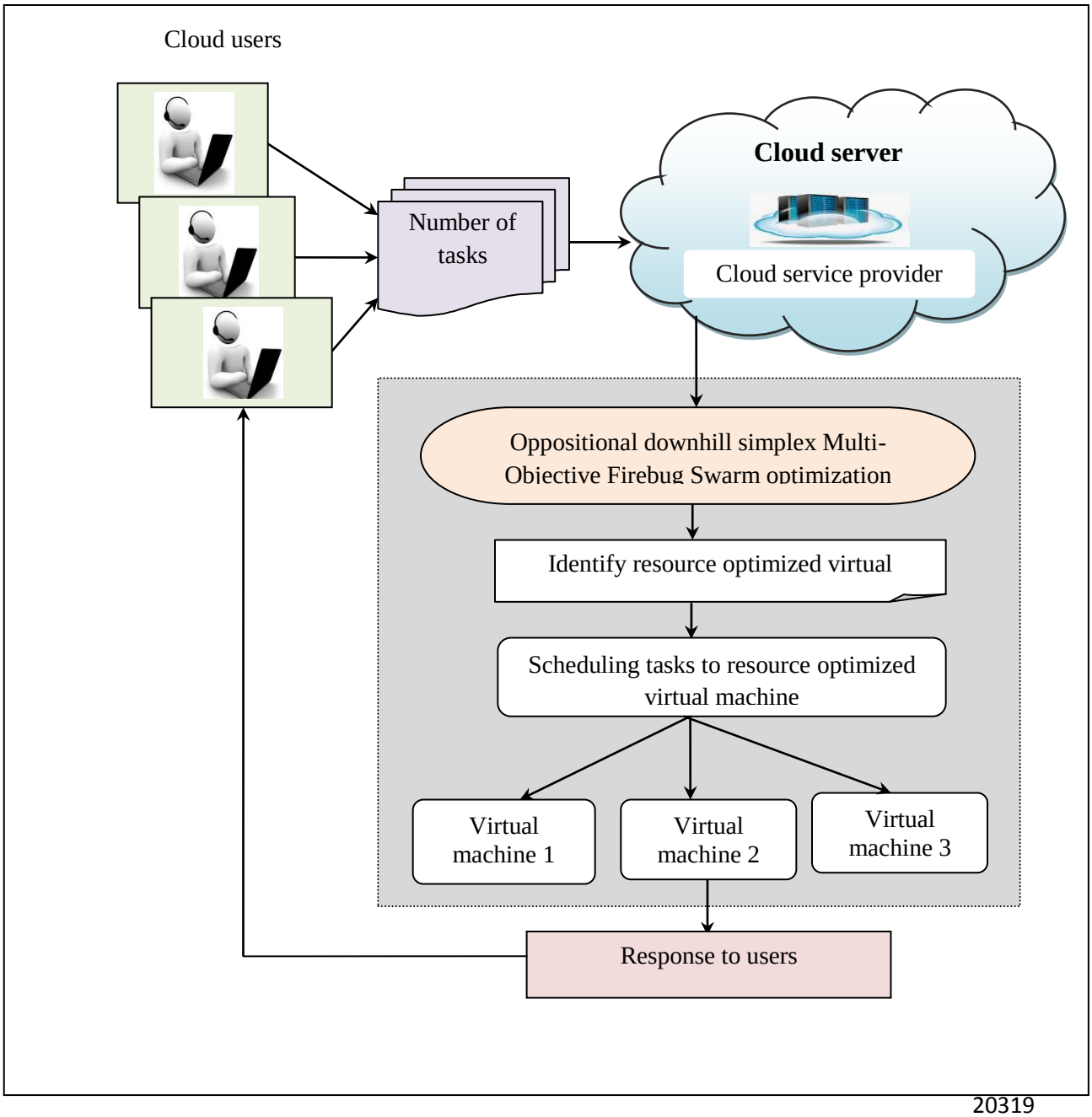

Figure 1 architecture of proposed ODSFSMROTS technique

Figure 1 illustrates the architecture diagram of the proposed ODSFSMROTS technique to perform resource-efficient task scheduling in the cloud. The proposed ODSFSMROTS technique comprises three entities such as users, cloud servers, and cloud service providers. A cloud user is a person who has a number of tasks submitted to a cloud server 'CS'. The cloud server is an important part of cloud technology and it is used for data computational and storage facilities. The cloud service provider offers various services according to the user's needs at anytime and anywhere.

First, the multiple cloud users' send their requests (i.e. tasks) to a cloud server. After receiving the request from the users, the cloud service provider is responsible for replying to the requests created by the users based on task scheduling. The main objective of scheduling is the optimal allocation of resources to particular tasks in a limited time to achieve high-performance computing and desirable quality of services.

The cloud server includes a collection of a virtual machines for both computational and storage facilities. The cloud service providers schedule the incoming user tasks arrived at the cloud server and manage the computing resources effectively by using Oppositional downhill simplex Multi-Objective Firebug Swarm optimization. In the proposed Firebug Swarm optimization, the Opposition-based learning concept is introduced to enhance the performance of task scheduling. Opposition-based Learning is a new concept that helps to identify the opposite relationship among entities for selecting the current best solution. The oppositional-based multiobjective Firebug Swarm optimization is a swarm intelligence that worked on the basis of population and stochastic search. Swarm intelligence–based algorithms are population-based algorithms developed based on the behavior of Firebug. The main behavior of Firebug is to find the best mates for reproduction in search space.

Here the Firebug is related to the number of virtual machines in a cloud server and identifying the best mates i.e., multiobjective functions namely memory consumption, bandwidth, energy, and CPU. Based on these resources, an optimal virtual machine (i.e. best

mates) is selected from the initial population. In this way, the resource-optimized virtual machine is identified. After finding the virtual machine, the service provider assigns the incoming tasks and responses to the end user with minimum time. This process minimizes the makespan of task scheduling. The process of the task scheduling and mathematical formulation of the proposed ODSFSMROTS technique is described in the subsequent sections.

3.1 Oppositional downhill simplex Multi-Objective Firebug Swarm optimization based resource efficient task scheduling in cloud

Oppositional downhill simplex Multi-Objective Firebug Swarm optimization-based task scheduling technique is introduced for managing a large number of tasks at cloud server with minimal makespan and efficient resource utilization. First, a system model of task scheduling in a cloud computing environment is presented. Followed by which, Oppositional Multi-Objective Firebug Swarm optimization is explained in detail.

3.1.2 System model

The cloud task scheduling problem is defined as scheduling the various tasks to various virtual machines (VMs) within a minimum time. Let us consider the cloud server (CS) which consists of many virtual machines$W_{m1,}W_{m2,}W_{m3,}\ldots W_{mn,}$. The number of incoming tasks $T_i \in T_1, T_2, T_3, \ldots . T_m$ where, $i = 1,23,..m$ generated by the cloud users $Cu_k \in Cu_1, Cu_2, Cu_3, \ldots . Cu_k$. The cloud service provider 'CSP' finds the resource effective virtual machines with multiple objective functions Energy(φ_E), Memory (φ_M), bandwidth(φ_B), CPU (φ_{CPU}) to allocate the incoming tasks 'T_i'.

3.1.3 Oppositional downhill simplex Multi-Objective Firebug Swarm optimization

The Oppositional downhill simplex Multi-Objective Firebug Swarm optimization process is explained in this section. The firebug optimization is a metaheuristic algorithm that is worked on the basis of the reproductive swarming behavior of firebugs. The significant process of firebug optimization is to search and find the best-fit reproductive partners as optimal solutions in a search space based on fitness estimation.

Initialize the population of 'n' Firebugs (i.e. virtual machine $W_{m1,}W_{m2,}W_{m3,}\dots W_{mn,}$) randomly in the search space. The population of Firebugs includes the product of the number of male bugs as well as female bugs.

$$P = M_b * F_b \quad (1)$$

Where, M_b indicates a male bug, F_b denotes a female bug, P represents the population of firebugs.

The oppositional learning method is applied to generate the opposite population along with the current population in search space to find the optimum solution.

$$P' = \alpha_i + \beta_i - P \quad (2)$$

From (2), P' indicates an opposite population generation, α_i and β_i symbolizes the minimum and maximum value of the dimensions in the current population'P'.

After the population initialization, the fitness is calculated for each female bug in the current population as well as the opposite population of the swarm. The fitness is an evaluation function used to find the optimum best solution (i.e. best virtual machine) in a given population for solving the desired problem (task scheduling).

The fitness function of each female bug is estimated based on multiple objective functions such as Energy(φ_E), Memory (φ_M), bandwidth(φ_B), CPU(φ_{CPU}) for finding the best optimal solution. The multi-objective function is a multiple criteria decision-making that includes two or more objective functions to be optimized simultaneously. Here, the multiple resources like Energy(φ_E), Memory (φ_M), bandwidth(φ_B), CPU(φ_{CPU}) are considered as multiple objective functions.

$$f\,(x) = \{\varphi_E,\ \varphi_M, \varphi_B, \varphi_{CPU}\} \quad (3)$$

Where, $f(x)$ denotes a multi-objective vector includes an objective functions$\varphi_E,\ \varphi_M, \varphi_B, \varphi_{CPU}$.

First, Energy is one of the major resources in the rapid growth of cloud data centers. The energy consumed by the virtual machine is estimated by the product of time consumed for processing the tasks and the average power in the cloud infrastructure

$$\varphi_E(W_m) = P_{cc} * time_S \quad (4)$$

Where, φ_E the energy of virtual machine (W_m), P_{cc} indicates power in watts and the $time_S$ denotes a time utilization is measured in seconds. Therefore, the energy availability of a virtual machine is calculated as follows,

$$\varphi_{E(avl)} = [\varphi_{E_{tt}} - \varphi_{E_{cc}}] \quad (5)$$

Where, $\varphi_{E(avl)}$ represents an energy availability of the virtual machine, $\varphi_{E_{tt}}$ denotes total energy, $\varphi_{E_{cc}}$ be the consumed energy.

Memory is the other significant resource of the virtual machine during the task scheduling processes. It is referred to as the amount of storage space required to process the tasks. The memory availability is measured to determine the amount of memory that is available for the allocation of tasks.

$$\varphi_{Mem(avl)} = [\varphi_{Mem(t)} - \varphi_{Mem(cc)}] \quad (6)$$

Where, $\varphi_{Mem(avl)}$ indicates the memory availability of the virtual machine, $\varphi_{Mem(t)}$ denotes a total memory capacity of the virtual machine and $\varphi_{Mem(cc)}$ denotes a utilized memory space of a virtual machine.

Next, bandwidth is a maximum capacity of a virtual machine to process and transfer tasks over a network connection.

$$\varphi_{B(avl)} = [\varphi_{B(t)} - \varphi_{B(cc)}\ ] \qquad (7)$$

Where, $\varphi_{B(avl)}$ indicates a bandwidth availability of the virtual machine, $\varphi_{B(t)}$ indicates the total bandwidth, $\varphi_{B(cc)}$ represents a consumed bandwidth.

Finally, the CPU time of the virtual machine is estimated as the number of CPUs allocated to the task scheduling process that has the minimum amount of burst time. The CPU burst time is the amount of time taken by a CPU to execute a certain task.

$$\varphi_{cpu(time)} = T_{cc}\ (\ Exec\_task\ )\ \ (8)$$

Where, $\varphi_{cpu(time)}$ denotes a CPU time of the virtual machine, T_{cc} indicates the time consumed for executing the task ($Exec\_task$). It is measured in milliseconds (ms).

After calculating the resources, fitness is measured with the help of the downhill simplex method to identify the best solution from the population. A downhill simplex method is a numerical method used to determine the minimum or maximum level of an objective function in a multidimensional space based on a relationship between one or more decision variables (i.e. resource of the virtual machine). Based on this downhill simplex method, the fitness of the female bug (i.e. virtual machine) is measured as given below,

$$F_{Bst} = ar\ g\ ma\ x\ [\varphi_{E(avl)},\ \varphi_{Mem(avl)}, \varphi_{B(avl)}]\ \&\&\ arg\ min\ [\varphi_{cpu(time)}] \qquad (9)$$

From (9), F_{Bst} represents fitness, $arg\ max$ refers to an argument of a maximum function, $arg\ min$ denotes an argument of a minimum function. Based on the fitness estimation, the different behaviors of firebugs are proposed in the following.

- **Mate selection**

The male bug only mates with the fittest female bug in its swarm population. The location of male bugs is initialized toward the location of the best female bug. In other words, each male bug position is always updated to be identical to the position of the best female in its

swarm. Therefore, the mate selection behavior of the algorithm is to select the best female among the current population as well as the opposite population based on fitness using a steady-state selection approach. By applying a steady state selection approach, current best female mates are selected with high fitness for the next process. Then some low-fitness female mates are removed in their place. This process significantly speeds up computation and minimizes the complexity.

$$S = \begin{cases} F_{Bst} > Th\,; & select\ \ the\ mates \\ Otherwise; & Reject\ the\ mates \end{cases} \quad (10)$$

Where S denotes a steady state selection outcome, Th indicates a threshold for each resource, F_{Bst} indicates a fitness. The selected few female mates are used for the next process.

- **Female position update**

After the selection process, the location of a few female mates' is updated from the current position. The positions of all the female bugs are stored in a single matrix and updated synchronously.

$$X_{ij}(F_b) = A(F_b) + \delta_1 \odot |X_n(F_b) - A(F_b)| + \delta_2 \odot |X_{best}(F_b) - A(F_b)| \quad (11)$$

Where, $X_{i+1}(F_b)$ updated position of the female mates', $A(F_b)$ position matrix of female mates' which contains i row and j column dimension,δ_1, δ_2 are the Acceleration coefficients, $X_n(F_b)$ denotes a position of the neighboring female mates', $X_{best}(F_b)$ denotes the best position of the female mates', $\odot$ Hadamard dot product.

- **Attraction of male bugs toward fittest female bugs**

Each male bug in the population is attracted towards the fittest females. This is done to avoid all-male bugs from being attracted to the same female bug leading to early convergence. The following update rule is used for the movement of male bugs towards the fittest female bug.

$$X_{ij}\,(M_b)' = X_{ij}(M_b) + \delta_3 \odot \left|X_{best}(F_b) - X_{ij}(M_b)\right| \quad (12)$$

Where, $X_{ij}\,(M_b)'$ updated position of the male bug which contains i row and j column dimension of the matrix, $X_{ij}(M_b)$ current position of male bugs which contains i row and j column dimension of the matrix, δ_3 denotes acceleration coefficients, $X_{best}(F_b)$ denotes the fittest position of the female bugs, $\odot$ denotes the Hadamard dot product.

- **Swarm cohesion**

The final behavior of the firebug optimization algorithm is swarm cohesion. The entire bug swarm moves collectively as a single unit, and individual bugs do not go away. Since the swarm moves as a single unit, individual bugs copy the movement direction of others towards the fittest female bug. A male bug does not move towards another male bug's location but it only copies its direction of movement towards good solutions, hence it reduces the early convergence to a local minimum.

$$X_{ij}(M_{ob}) = X_{ij}(M_{ob}) + \delta_4 \odot \left|X_{best}(F_b) - X_{ij}(M_{ob})\right| \quad (13)$$

Where, $X_{ij}(M_{ob})$ updated position of the other male bugs, $X_{ij}(M_{ob})$ current position of male bugs which contains i row and j column dimension of the matrix, δ_4 denotes acceleration coefficients, $X_{best}(F_b)$ denotes the fittest position of the female bugs, $\odot$ Hadamard dot product. This entire process is iterated until the convergence criteria are satisfied. The flow chart of the optimization algorithm is given below,

Section A-Research paper
ISSN2063-5346

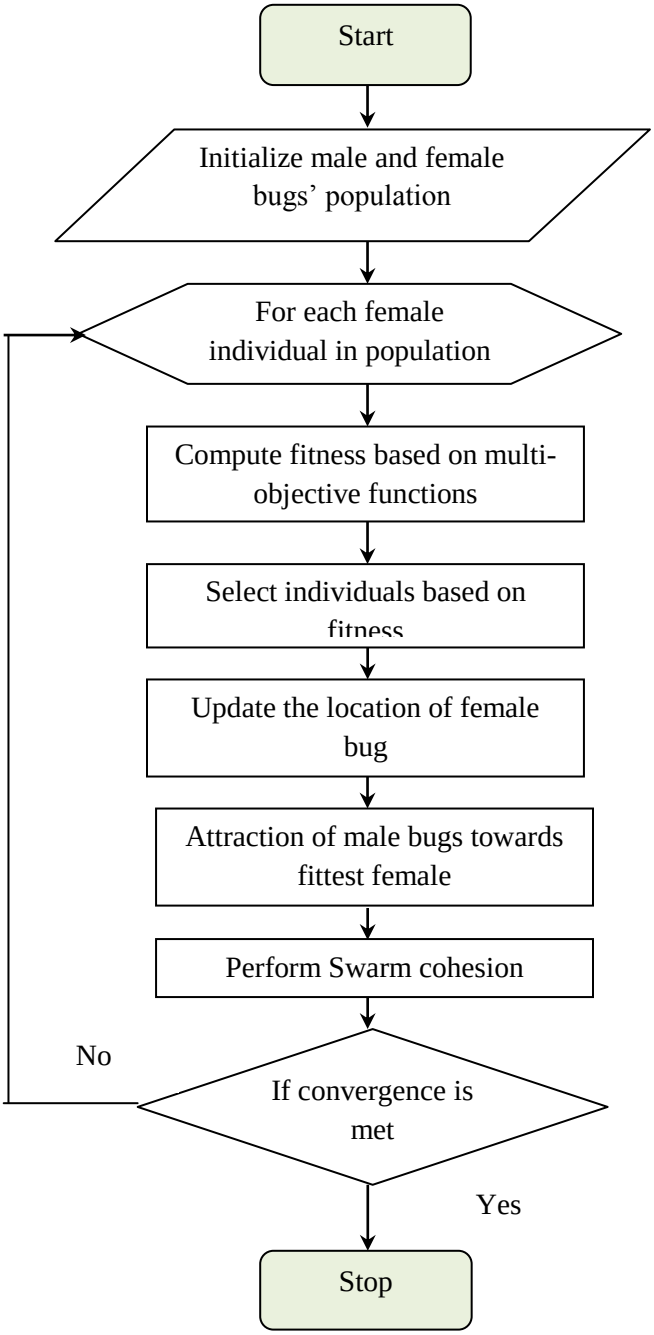

Figure 2 flowchart of Oppositional downhill simplex Multi-Objective Firebug Optimization

Figure 2 illustrates the flowchart of the oppositional downhill simplex multi-objective firebug optimization to find the best firebug (i.e. virtual machine). After finding the resource-optimized virtual machine, the incoming user tasks are scheduled for that machine. Finally, the

cloud service provider responds to the users with minimum time. The ODSFSMROTS algorithm is described as given below,

// Algorithm 1: Oppositional downhill simplex Multi-Objective Firebug Swarm Resource Optimized Task Scheduling

Input: cloud server (CS), number of virtual machines $W_{m1,}W_{m2,}W_{m3,}\dots W_{mn,}$, number of incoming tasks $T_i \in T_1, T_2, T_3, \dots . T_m$, cloud users $Cu_k \in Cu_1, Cu_2, Cu_3, \dots . Cu_k$, cloud service provider 'CSP'

Output: Increase the task scheduling efficiency

Begin

1. Initialize the population of the virtual machine $P = W_{m1,}W_{m2,}W_{m3,}\dots W_{mn,}$
2. Initialize opposite populations P'
3. **for each** $W_{mi,}$ in P and P'
4. **Compute multiple** resources $f\ (x) = \{\varphi_E,\ \varphi_M, \varphi_B, \varphi_{CPU}\ \}$
5. Calculate the fitness 'Y'
6. **End for**
7. Merge the populations P and P'
8. Select current best 'n' virtual machines
9. ***if*** **(**$F_{Bst} > Th$**)** **then**
10. Select current best 'n' virtual machines
11. **else**
12. Remove other virtual machines which less fitness
13. **End if**
14. **While (t < max_ iteration)**
15. **Update the female firebug position** '$X_{ij}(F_b)$ ' using **(11)**
16. Attraction of male bugs towards fittest female bugs '$X_{ij}\ (M_b)'$' using (12)
17. Perform Swarm cohesion '$X_{ij}(M_{ob})$' using (13)
18. **t= t+1**
19. **end while**

20. Obtain the best optimal virtual machine

21. Schedule the tasks to the best optimal virtual machine

End

Algorithm 1 describes the step-by-step process of task scheduling using an oppositional downhill simplex multi-objective firebug swarm optimization. For each incoming user task, the cloud server starts to perform the task scheduling. The population of firebugs (virtual machine) and the opposite swarm populations are randomly initialized in the search space. For each female firebug in the population, the proposed optimization technique calculates the fitness with multiple resources such as bandwidth, memory, energy, and CPU time of the virtual machine. Followed by, virtual machines in the two populations are merged. Then the steady state selection approach is applied for choosing the current best 'n' virtual machines for finding the global optimum solution to minimize the complexity. Then the female firebug position gets updated based on their fitness. Followed by, the attraction of male bugs towards fittest female bugs is updated. Finally, the other male bugs copy the direction of other male bugs toward the fittest female bug. Finally, the global best resource-efficient virtual machine is selected for processing the user-requested tasks. The entire process gets iterated until the algorithm reaches the maximum iteration. In the above algorithms', 't' represents the iteration count of the algorithm. Finally, the cloud server allocates the high-priority tasks to the resource optimal virtual machine, and the service provider responds to the users with minimum time.

4. EXPERIMENTAL SETTINGS

Experimental evaluations of the proposed ODSFSMROTS technique and existing methods namely TSMGWO [1] and [2] are implemented using Java language with the CloudSim network simulator. The Personal Cloud Datasets http://cloudspaces.eu/results/datasets is used for task scheduling in the cloud. The dataset includes 17 attributes (i.e. columns) and 66245 instances. The 17 attributes are row id, account id, file size (i.e. task size), operation_time_start, operation_time_end, time zone, operation_id, operation type, bandwidth trace, node_ip, node_name, quoto_start, quoto_end, quoto_total (storage capacity), capped, failed and failure info. Among 17 attributes, two attributes namely time zone and capped are not used. The remaining attributes are used for task scheduling to multiple virtual machines in the cloud server.

5. PERFORMANCE EVALUATION UNDER VARIOUS PARAMETERS

In this section, the performance analysis of the proposed ODSFSMROTS technique and existing methods namely TSMGWO [1] and MOABCQ [2] are discussed with different metrics such as task scheduling efficiency, false-positive rate, makespan, and memory consumption. The performances of the three methods are discussed with the help of a table and graphical representation.

Task scheduling efficiency: it is measured as the number of incoming user tasks are correctly scheduled to the resource-efficient virtual machine. The formula for calculating the task scheduling efficiency is given below,

$$Task_{SE} = \left(\frac{n_{CS}}{n}\right) * 100 \qquad (14)$$

Where, $Task_{SE}$ indicates a task scheduling efficiency, n_{CS} indicates the number of tasks correctly scheduled, n denotes the number of tasks. The task scheduling efficiency is measured in terms of percentage (%).

Makespan: it is defined as the amount of time taken to schedule the user-requested tasks to virtual machines. The Makespan is mathematically calculated as follows,

$$M = n * t\,(SST) \quad (15)$$

Where M indicates a makespan, n represents the number of tasks, t indicates a time, SST denotes the scheduling of one task. Makespan is measured in terms of milliseconds (ms).

Throughput: it refers to the number of user-requested tasks executed within a unit time in the cloud environment. The time is measured in seconds.

$$TPT = \left[\frac{Number\ of\ tasks\ executed}{time\ (sec)}\right] \qquad (16)$$

Where TPT denotes a throughput ad it measured in terms of tasks per second (tasks/sec).

Memory Consumption: it is measured as an amount of storage space consumed by the algorithm to schedule the user-requested tasks in the cloud environment. The overall memory consumption is expressed as follows,

$$Cons_{Mem} = n * Mem\ (SST) \qquad (17)$$

Where $Cons_{Mem}$ indicates a memory consumption, '$Mem\ (SST)$' denotes the memory consumed to schedule the single-user requested tasks and 'n' indicates a total number of user-requested tasks. The memory consumption is calculated in the unit of megabytes (MB).

Table I comparison of Task Scheduling Efficiency

Number of tasks	Task Scheduling Efficiency (%)		
	ODSFSMROTS	**TSMGWO**	**MOABCQ**
500	96	92	90
1000	95.8	91.5	89
1500	95.33	91.33	88.33
2000	94.5	90.5	87.75
2500	94.36	89.2	86.8
3000	94	88.33	85
3500	93.85	87.71	84.28
4000	93.5	87	83.75
4500	92.88	86.88	82.66
5000	92.2	86	82

Table 1 reports the performance analysis of task scheduling efficiency of three different methods namely the ODSFSMROTS technique and TSMGWO [1] and MOABCQ [2] versus a number of user tasks taken in the rages from 500 to 5000. For each method, ten different results are observed with respect to a number of tasks. The observed results demonstrate that the ODSFSMROTS observed superior performance than the other existing methods. This is inferred from the statistical analysis. Let us consider the experimental is conducted with 500 tasks for measuring the efficiency of three methods. By applying the ODSFSMROTS, the task scheduling efficiency was observed to be 96%, whereas the efficiency using, [1], and [2] were found to be 92%, and 90% respectively. For each method, ten probable simulation results were inferred and the results obtained from the proposed method were compared to the results of the existing methods. The average of ten simulation comparison results shows that the task scheduling efficiency using ODSFSMROTS is improved by 6% compared to [1], and 10% compared to [2] respectively.

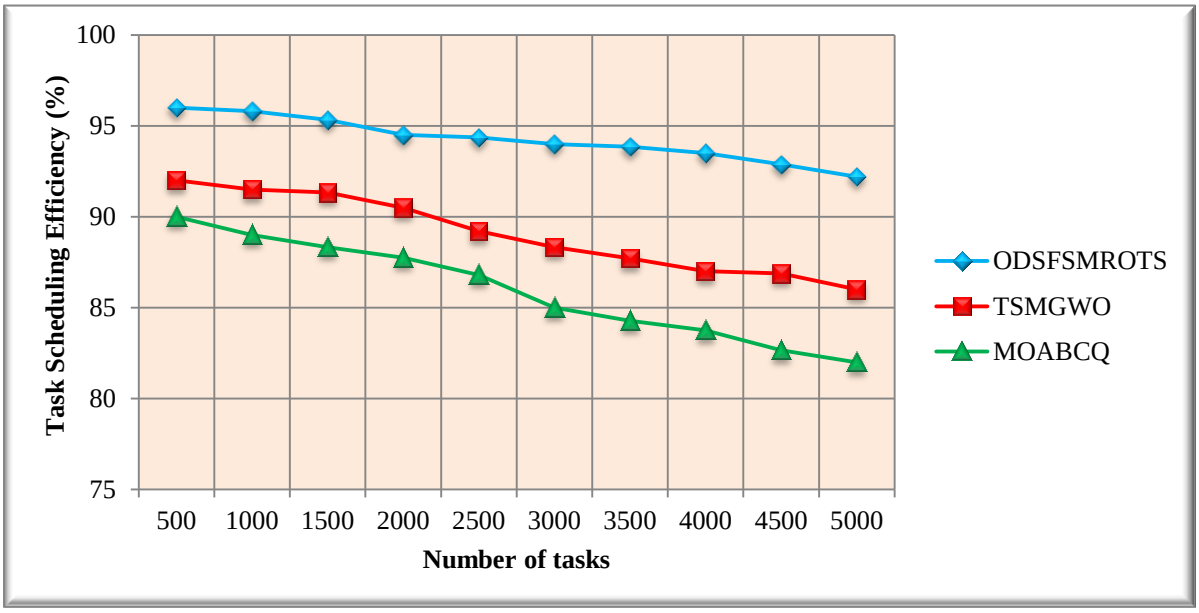

Figure 3 comparison of task scheduling efficiency

Figure 3 given above displays the graphical representation of task scheduling efficiency time versus distinct numbers of tasks arrived at the cloud server. From the above graphical representation, it is inferred that the number of tasks is taken as input on the horizontal axis whereas the performance of task scheduling efficiency is observed on the vertical axis. As shown in the graph, the ODSFSMROTS technique provides superior performance upon

comparison with the other existing task scheduling methods. This improvement is said to be attained via Oppositional downhill simplex Multi-Objective Firebug Swarm optimization. By applying this algorithm, the cloud server starts to find the resources of the virtual machine based on bandwidth, memory, energy, and CPU time. Then fitness is measured for each virtual machine with the help of the downhill simplex method. After that, the steady state selection approach is applied to find the current best and finally, the global best fittest virtual machine is identified based on position updates. Then they arrived incoming tasks are correctly scheduled to the optimal virtual machine with higher efficiency.

Table 2 comparison of Makespan

Number of tasks	Makespan (ms)		
	ODSFSMROTS	**TSMGWO**	**MOABCQ**
500	22	25	27.5
1000	25	27	29
1500	30	33	35.25
2000	32	34	36
2500	35	37.5	40
3000	37.5	39	42
3500	38.5	40.25	43.75
4000	40	42	44
4500	42.75	45.45	47.25
5000	45	50	52

Table 2 given above lists the experimental analysis of Makespan versus different numbers of user-requested tasks obtained as input ranging between 500 and 5000. From the above tabulation results, increasing the number of tasks, the Makespan gets increased using all three task scheduling methods. However, the proposed ODSFSMROTS technique consumes a lesser Makespan for scheduling numerous tasks. Let us consider experiment is performed with 500 tasks, the time consumed for scheduling the tasks was observed to be $22ms$ using the proposed ODSFSMROTS technique. Similarly, the time consumed in task scheduling using

TSMGWO [1] and MOABCQ [2] were found to be $25ms$, $27.5ms$ respectively. Experimental was performed for 10 different runs along with a different number of tasks. The performance results of ODSFSMROTS are compared to the results obtained by using existing methods. The average values were obtained for making a comparative analysis. The comparative analysis, shows that the ODSFSMROTS minimizes the Makespan by 7% and 13% than the existing [1] [2] respectively.

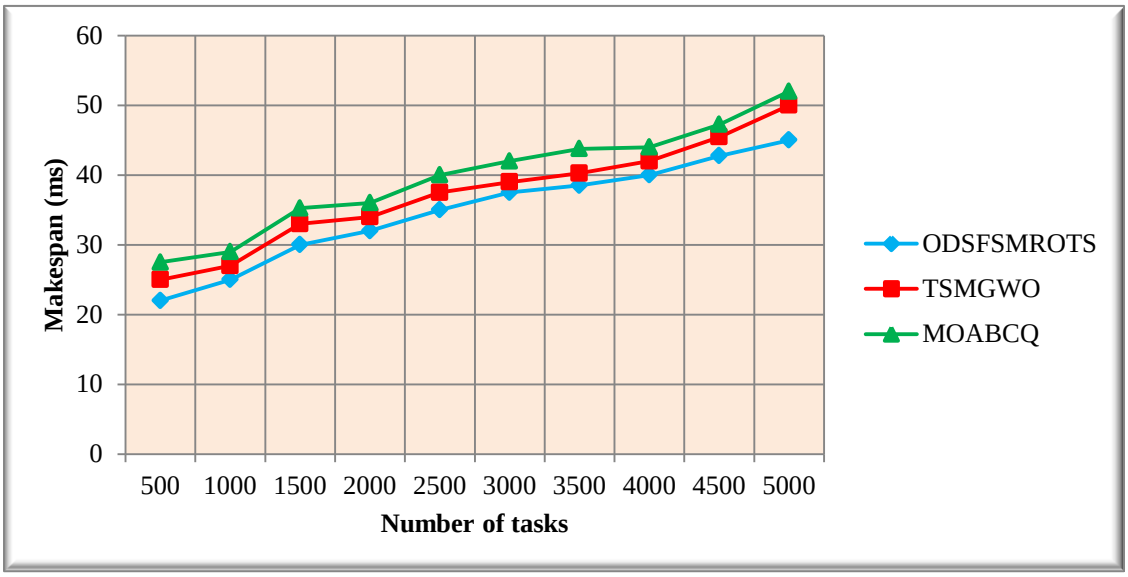

Figure 4 Graphical representation of Makespan

Figure 4 given above displays the graphical illustration of makespan versus 5000 distinct numbers of tasks obtained from the cloud users. From the above graphical representation, it is inferred that ODSFSMROTS decreases the makespan upon comparison with the other existing methods. This improvement is said to be achieved by fining the resource-optimized virtual with the help of the Multi-Objective Firebug Swarm optimization. The steady-state selection approach is applied to a Multi-Objective optimization for selecting the best virtual machine and another virtual machine that has low fitness is removed from the entire population. This helps to minimize the overall time consumption of selecting the optimal virtual machine. After finding the virtual machine, the task is scheduled to that optimal virtual machine with minimum time.

Section A-Research paper
ISSN2063-5346

Table 3 Comparison of Throughput

Number of tasks	Throughput (tasks/sec)		
	ODSFSMROTS	**TSMGWO**	**MOABCQ**
500	220	170	147
1000	432	360	275
1500	518	430	378
2000	712	623	563
2500	847	745	674
3000	936	869	785
3500	1020	965	898
4000	1150	1050	980
4500	1250	1180	1075
5000	1320	1260	1146

Table 3 given above lists the comparative performance analysis of throughput versus the number of tasks taken as input in the ranges between 500 and 5000. From the above-tabulated results, a different analysis of throughput is observed using all three methods, the ODSFSMROTS technique and TSMGWO [1], and MOABCQ [2]. Among the three methods, ODSFSMROTS achieves higher throughput when compared with the other existing methods. This is inferred from the statistical analysis. Let us consider 500 tasks for conducting the experiment. By applying the ODSFSMROTS, the 220 tasks are executed per one second. Whereas the 170 tasks and 147 tasks are executed per one second using, [1], and [2] respectively. For each method, ten feasible results were inferred and compared. The average of ten simulation comparison results shows that the performance of throughput using ODSFSMROTS was increased by 13% compared to [1], and 28% compared to [2] respectively.

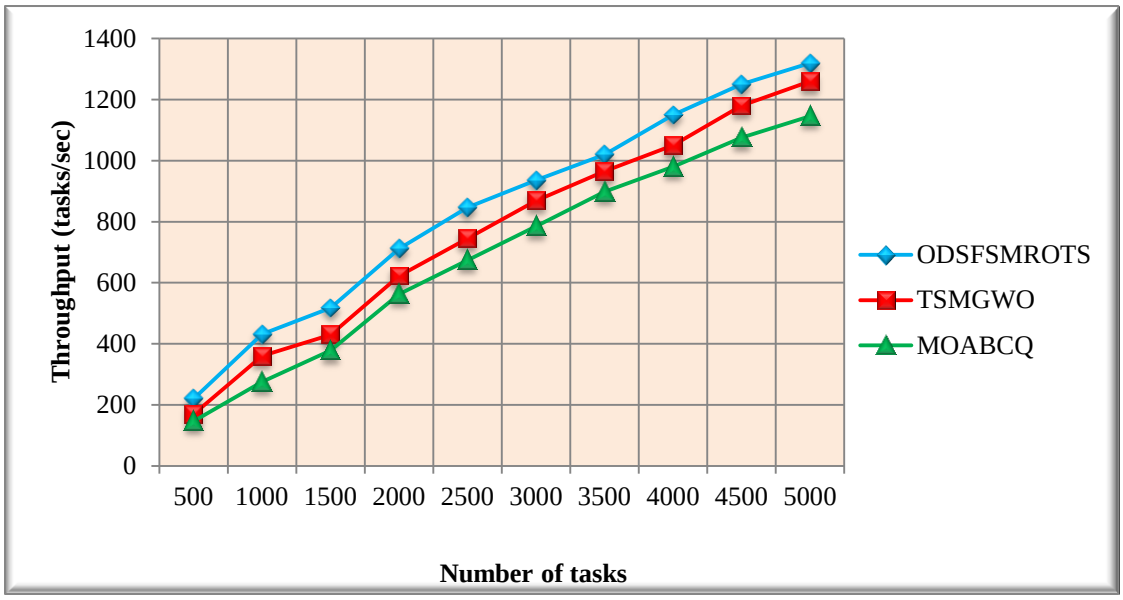

Figure 5 Graphical representation of throughput

Figure 5 given above illustrates the graphical representation of throughput using three different methods, namely, the ODSFSMROTS technique and TSMGWO [1], and MOABCQ [2]. As shown in the above figure 5, the horizontal axis refers to the number of tasks arrived in the cloud server and the vertical axis refers to the throughput. From the above results, the proposed ODSFSMROTS technique attained better performance of throughput than the other existing methods. The reason for the improvement is to find a resource-efficient virtual machine by applying a Multi-Objective Firebug Swarm optimization. The proposed optimization algorithm efficiently finds the best virtual machine which has higher energy availability, bandwidth availability, and lesser CPU time. The selected resource-efficient virtual machines frequently execute the tasks within the given time.

Table 4 comparison of memory consumption

Number of tasks	Memory consumption (MB)		
	ODSFSMROTS	TSMGWO	MOABCQ
500	27.5	30	32
1000	33	36	39
1500	35.25	37.5	40.5

2000	43	44.8	48
2500	45.5	47.5	50
3000	51.6	54	56.7
3500	56.7	59.15	63
4000	58	62	64
4500	62.1	65.25	67.5
5000	64	67	70

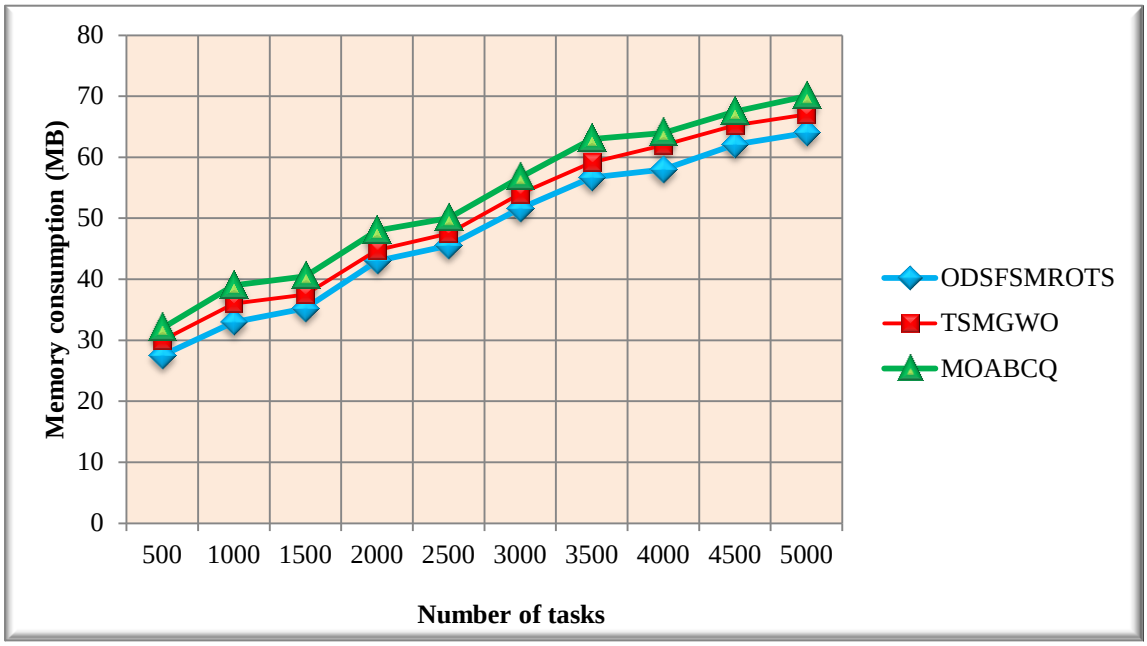

Figure 6 Graphical representation of memory consumption

Finally table 4 and figure 6 given above provides the performance of memory consumption. From the above figure horizontal axis represents the number of user tasks ranging between 500 and 5000. On the other hand, the vertical axis indicates the memory consumption measured in terms of Megabytes (MB). Also from the above graphical representation, the memory consumption is found to be directly proportionate to the number of tasks provided as input. In other words, increasing the number of tasks causes an increase in memory consumption. However, the experiment is conducted with 500 tasks and observed memory consumption of the ODSFSMROTS technique and TSMGWO [1] and MOABCQ [2] are $27.5MB, 30MB,$ and $32MB$ respectively. From these results, the memory consumption using ODSFSMROTS is said

to be comparatively lesser than the existing methods. The reason behind the improvement was due to the selecting the better memory availability of the virtual machine. The algorithm consumes lesser memory space for scheduling the numerous incoming tasks arrived at the cloud server for multiple resources with lesser memory consumption Finally, the overall performance of memory consumption is reduced using the ODSFSMROTS technique by 6% compared to [1], 11% compared to [2] respectively.

6. CONCLUSION

Task scheduling is a significant challenge that affects the overall performance of the cloud computing environment. The major contribution of this paper is to enhance the Task scheduling efficiency algorithm by proposing a novel technique named ODSFSMROTS. It concentrates on providing a solution for resource-optimized task scheduling and minimizing the makespan. In the ODSFSMROTS technique, the user-requested tasks are sent to the cloud server. Then, the server uses the Oppositional downhill simplex Multi-Objective Firebug Swarm Optimization algorithm to find the optimal virtual machines that have better resource utilization with help of the downhill simplex method. After that, the multiple tasks are assigned to selected virtual machines for executing the numerous tasks. Experimental assessment is performed using a personal cloud dataset. Quantitative analysis is performed using the ODSFSMROTS method and the other existing methods with different metrics such as task scheduling efficiency, makespan, throughput, and memory consumption. The obtained results indicate that our ODSFSMROTS technique provides improved performance in terms of higher task scheduling efficiency, throughput and lesser makespan, and memory consumption when compared to other existing methods.

REFERENCES

[1] Deafallah Alsadie, "TSMGWO: Optimizing Task Schedule Using Multi-Objectives Grey Wolf Optimizer for Cloud Data Centers", IEEE Access, Volume 9, 2022, Pages 37707 – 377252, DOI: 10.1109/ACCESS.2021.3063723

[2] Boonhatai Kruekaew; Warangkhana Kimpan, "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee

Colony Algorithm With Reinforcement Learning", IEEE Access , Volume 10, 2022, Pages 17803 – 17818. DOI: 10.1109/ACCESS.2022.3149955

[3] LiWei Jia , Kun Li, and Xiaoming Shi, "Cloud Computing Task Scheduling Model Based on Improved Whale Optimization Algorithm", Wireless Communications and Mobile Computing, Hindawi, Volume 2021, December 2021, Pages 1-13. https://doi.org/10.1155/2021/4888154

[4] Hadeer Mahmoud; Mostafa Thabet; Mohamed H. Khafagy; Fatma A. Omara, "Multiobjective Task Scheduling in Cloud Environment Using Decision Tree Algorithm", IEEE Access ,Volume: 10, 2022, Pages 36140 – 36151. DOI: 10.1109/ACCESS.2022.3163273

[5] Rambabu Medara, Ravi Shankar Singh, Amit, "Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization", Simulation Modelling Practice and Theory, Elsevier, Volume 110, 2021, Pages 1-16. https://doi.org/10.1016/j.simpat.2021.102323

[6] Bilal H. Abed-alguni and Noor Aldeen Alawad, "Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments", Applied Soft Computing, Elsevier, Volume 102, 2021, Pages 1-15. https://doi.org/10.1016/j.asoc.2021.107113

[7] Fahd Alhaidari and Taghreed Zayed Balharith, "Enhanced Round-Robin Algorithm in the Cloud Computing Environment for Optimal Task Scheduling", Computers, Volume 10, Issue 5, 2022, Pages 1-27. https://doi.org/10.3390/computers10050063

[8] Xueying Guo, "Multi-objective task scheduling optimization in cloud computing based on fuzzy self-defense algorithm", Alexandria Engineering Journal, Elsevier, Volume 60, Issue 6, December 2021, Pages 5603-5609. https://doi.org/10.1016/j.aej.2021.04.051

[9] Wanneng Shu, Ken Cai, Neal Naixue Xiong, "Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing", Future Generation Computer Systems, Elsevier, Volume 124, 2021, Pages 12-20. https://doi.org/10.1016/j.future.2021.05.012

[10] Sudheer Mangalampalli, Sangram Keshari Swain & Vamsi Krishna Mangalampalli, "Prioritized Energy Efficient Task Scheduling Algorithm in Cloud Computing Using Whale Optimization Algorithm", Wireless Personal Communications, Springer, 2021, Pages 1-17. https://doi.org/10.1007/s11277-021-09018-6

[11] Deafallah Alsadie, "A Metaheuristic Framework for Dynamic Virtual Machine Allocation With Optimized Task Scheduling in Cloud Data Centers", IEEE Access, Volume 9, 2021, Pages 74218 – 74233. DOI: 10.1109/ACCESS.2021.3077901

[12] Avinab Marahatta, Sandeep Pirbhulal, Fa Zhang, Reza M. Parizi, Kim-Kwang Raymond Choo, Zhiyong Liu, "Classification-based and Energy-Efficient Dynamic Task Scheduling Scheme for Virtualized Cloud Data Center", IEEE Transactions on Cloud Computing , Volume 9, Issue 4, 2021, Pages 1376 – 1390. DOI: 10.1109/TCC.2019.2918226

[13] Merve Nur Aktan and Hasan Bulut, "Metaheuristic task scheduling algorithms for cloud computing environments", Concurrency Computation Practice Experience, Wiley, Volume 34, Issue 9, 2022, Pages 1-13. https://doi.org/10.1002/cpe.6513

[14] Heba M. Eldesokey, Saied M. Abd El-atty, Walid El-Shafai, Mohammed Amoon, Fathi E. Abd El-Samie, "Hybrid swarm optimization algorithm based on task scheduling in a cloud environment", Volume 34, Issue 13, 2021, Pages 1-14. https://doi.org/10.1002/dac.4694

[15] Ya Zhou and Xiaobo Jiao, "Knowledge-Driven Multi-Objective Evolutionary Scheduling Algorithm for Cloud Workflows", IEEE Access, Volume 10, 2021, Pages 2952 – 2962. DOI: 10.1109/ACCESS.2021.3139137

[16] Lilu Zhu, Kai Huang, Yanfeng Hu, Xianqing Tai, "A Self-Adapting Task Scheduling Algorithm for Container Cloud Using Learning Automata", IEEE Access , Volume 9, 2021, Pages 81236 – 81252. DOI: 10.1109/ACCESS.2021.3078773

[17] Dibyendu Mukherjee, Shivnath Ghosh, Souvik Pal, Ayman A. Aly, Dac-Nhuong Le, "Adaptive Scheduling Algorithm Based Task Loading in Cloud Data Centers", 2022, IEEE Access, Volume 10, Pages 49412 – 49421. DOI: 10.1109/ACCESS.2022.3168288

[18] Sheng Wang, Zhijun Ding, Changjun Jiang, "Elastic Scheduling for Microservice Applications in Clouds", IEEE Transactions on Parallel and Distributed Systems, Volume 32, Issue 1, 2021, Pages 98 – 115. DOI: 10.1109/TPDS.2020.3011979

[19] Shida Lu, Rongbin Gu, Hui Jin, Liang Wang, Xin Li, Jing Li, "QoS-Aware Task Scheduling in Cloud-Edge Environment", IEEE Access, Volume 9, 2021, Pages 56496 – 56505. DOI: 10.1109/ACCESS.2021.3072216

[20] Hojjat Emami, "Cloud task scheduling using enhanced sunflower optimization algorithm", ICT Express, Elsevier, Volume 8, Issue 1, 2022, Pages 97-100. https://doi.org/10.1016/j.icte.2021.08.001