Advances in Dynamical Systems and Applications.
ISSN 0973-5321, Volume 16, Number 2, (2021) pp. 1303-1320

https://www.ripublication.com/adsa.htm

Oppositional Learnt Multi-Objective Dragonfly Resource Optimized Dynamic Task Scheduling in Cloud

S.Tamilsenthil [1] and Dr. A. Kangaiammal[2]

Ph.D Research Scholar (Part Time), PG & Research Dept of Computer Science, Government Arts College (Autonomous), Salem-7, India.

Assistant Professor, Dept of Computer Science, Padmavani Arts and Science College for Women, Salem-11, India.

Assistant Professor, Dept of Computer Applications, Government Arts College (Autonomous), Salem-7, India.

Abstract

Cloud Computing is the service provider model in which users submit their requests to the server for execution. The cloud server schedules a variety of requests and handles the resources efficiently. Scheduling becomes a significant task in the cloud framework. Many scheduling algorithms have been proposed in literature, but the time consumption and efficiency for scheduling is a major problem. Therefore, an efficient algorithm is needed to improve scheduling efficiency and reduced time consumption. A novel technique called Oppositional Learnt Multi-objective Dragonfly Resource Optimized Dynamic Task Scheduling (OLMDRODTS) is proposed with the aim of increasing the efficiency and minimizing the time consumption. The user dynamically submits multiple heterogeneous tasks to the cloud server. The task scheduler in the cloud receives the incoming tasks and determines the resource-efficient virtual machine for the scheduling process. The Oppositional Learnt Multi-objective Dragonfly Optimization is used to find an efficient virtual machine based on the different resources such as CPU time, memory, bandwidth, and energy. By applying the optimization technique, first, the population of the virtual machine is initialized. Based on the multiple objective functions, the fitness is computed for each virtual machine in the cloud server. The virtual machine with maximum resource availability is chosen as a global optimum than the others. Then the scheduler assigns the user-requested tasks to the resource-efficient virtual machine to minimize the job response time and overhead. Experimental evaluation is conducted in CloudSim simulator using a personal cloud dataset with different performance metrics such as task scheduling efficiency, false-positive rate, computation

overhead, and memory consumption with respect to the number of user-requested tasks. The observed results indicate that the proposed OLMDRODTS technique outperforms well for achieving the higher task scheduling efficiency with lesser computation overhead, false-positive rate as well as memory consumption.

Keywords: Cloud; Task Scheduling; Resource Optimization; Oppositional Learning; Multi-objective Dragonfly Optimization.

1. INTRODUCTION

Cloud Computing is a type of distributed technology that provides various computational services such as storage services and other web-based applications. With the increasing operation of data centers around the world, cloud computing is a significant paradigm for large-scale applications. However, these cloud environments face lots of challenges including resource optimized task scheduling.

A Chaotic Squirrel Search Algorithm (CSSA) was introduced in [1] for multitask scheduling with efficient resource utilization. But the designed CSSA optimization algorithm failed to find the greatest compromise solution with minimum time. A new Harmony-Inspired Genetic Algorithm (HIGA) was designed in [2] for energy-efficient task scheduling on the cloud data center. Through the algorithm reduces the makespan and execution overhead, the scheduling efficiency was not improved.

An improved task scheduling method was introduced in [3] to increase the efficiency and minimize the resource utilization. But the optimization technique was not applied to solve the multi-objective problems. An Improved Particle Swarm Optimization (PSO) algorithm was designed in [4] to minimize the job completion time. The designed algorithm reduces the makespan but the fault tolerance was not achieved.

An Estimation of Distribution Algorithm and GA (EDA-GA) was developed in [5] to efficiently minimize the task completion time and improve the load balancing capability. However, the algorithm failed to solve multi-objective problems. An Enhanced Multi-Verse Optimizer (EMVO) algorithm was developed in [6] to improve the task scheduling efficiency with better resource utilization. However, the algorithm failed to perform dynamic task scheduling with multiple tasks.

An efficient map-reduce framework and Genetic Algorithm based Whale Optimization Algorithm (GA-WOA) was introduced in [7] for efficient task scheduling with minimum time. However, the energy-based task scheduling remained unaddressed. The Metaheuristics Whale Optimization Algorithm (WOA) was developed in [8] for task scheduling with a multi-objective optimization to improve the performance of a cloud system. But the algorithm failed to reduce the scheduling overhead in the presence of large workloads. A Particle Swarm Optimization (PSO) algorithm based task scheduling was performed in [9] to reduce the makespan. But, the proposed PSO-based scheduler failed to schedule the workflow with multiple optimization objectives in the cloud environment.

A Multi-Criteria Decision-Making method was introduced in [10] for energy-efficient task-scheduling with lesser resource utilization. The designed method failed to analyze the efficiency of the proposed task scheduling method. A stochastic approximation approach was introduced in [11] for task scheduling to decrease the response time as well as makespan and also increases resource efficiency. However, the overhead was not minimized.

AQ-learning based Task Scheduling approach was developed in [12] for energy-efficient cloud computing and minimizing the task response time. But the approach failed to further evaluate the scheduling approach in large scale cloud environments. A Crow–Penguin Optimizer was introduced in [13] for task scheduling based on multiple objective functions to minimize the execution time and makespan. But the scheduling efficiency was not improved.

A Directed Acyclic Graph (DAG) based tasks scheduling was performed in [14] to reduce the overall makespan and task execution time. But the approach failed to perform the dynamic scheduling and it also failed to consider the energy consumption of task scheduling. A Cuckoo Search (CS) and Particle Swarm Optimization were developed in [15] to schedule the tasks to a virtual machine with lesser makespan. However, the algorithm failed to optimize multiple resources.

A Multi-Faceted Optimization Scheduling Framework (MFOSF) was developed in [16] for task scheduling with lesser resource utilization. But the framework was not efficient to perform the scheduling with minimum overhead. A Resource-Constrained Task Scheduling algorithm was designed in [17]. However, the algorithm failed to consider the resource utilization and energy consumption of task scheduling in the cloud.

An Energy-Aware, Time, and Throughput Optimization Heuristic Algorithm was introduced in [18] for cloud environments to address the multi-objective optimization. But the algorithm failed to support the memory and bandwidth for solving the multi-objective optimization.

A new Hybrid Bio-Inspired Algorithm was introduced in [19] for task scheduling with lesser resource utilization. The designed algorithm failed to perform dynamic scheduling in the cloud environment. In order to dynamically schedule the tasks with minimum execution time, a deep Reinforcement Learning Architecture (RLTS) was introduced in [20]. However, the architecture failed to consider the multi-objective task scheduling problem.

The above-said issues are addressed by introducing a novel technique called OLMDRODTS. The overall contribution of the OLMDRODTS is summarized as given below,

- To improve the task scheduling efficiency, the OLMDRODTS technique is introduced for finding the resource-efficient virtual machine in the cloud. The oppositional learning concept is applied in the multi-objective dragonfly optimization for selecting the optimum virtual machine through the fitness evaluation based on the CPU time, bandwidth, energy, and memory. Then the

scheduler dynamically assigns the incoming tasks to the global best virtual machine for completing the certain task.

- To reduce the false-positive rate, Multi-Objective Dragonfly optimization uses the oppositional learning concept for generating the opposite population with the current population to achieve fault tolerance. This helps to find the global optimum for scheduling the tasks.
- To minimize the computation overhead, the OLMDRODTS technique uses the ranking method to find the local optimum from the current and opposite population of the virtual machine. The local optimum virtual machine is selected from the population-based fitness calculation. This helps to minimize the time taken to find the global optimum as well as task scheduling time.
- Finally, an extensive experiment is conducted to evaluate the performance of our OLMDRODTS technique and related works. The observed result demonstrates that our OLMDRODTS technique outperforms well than the other optimization methods

1.1. *Organization of Paper*

The paper is organized into five different sections. Section 2 describes the OLMDRODTS technique for resource-efficient task scheduling in the cloud. In section 3, experimental evaluation is conducted with a dataset and the performance of various metrics is discussed in section 4. Finally, section 5 provides the conclusion of the work.

2. METHODOLOGY

In cloud computing, task scheduling is a process in which user transmitted tasks that needs to be executed to the available resources through the internet. To satisfy the user requirements and application requirements, the cloud service provider uses a resource allocation policy for particular tasks. The cloud service provider needs to know about how much as well as which types of resources, the status of each resource required to complete a particular task. When the user submits the task, the cloud scheduler identifies the best virtual machine with an optimized resource to fit the task from the available resources. Then the allocation of resources for the submitted tasks is done through the task scheduler. Finally, the task is executed with the minimum response time. Based on this concept, a novel OLMDRODTS technique is proposed.

2.1. *Network Model*

The cloud computing architecture is designed based on the following network model. The architecture comprises ‘n’ independent tasks,

$`T = \{T_1, T_2, \dots, T_n\}$ dynamically generated from the various users

`$U = \{u_1, u_2, \ldots, u_m\}$ arrived in the queue and task scheduler 'TS' schedules a set of 'b' virtual machines

`$Vm = \{Vm_1, Vm_2, \ldots, Vm_b\}$ in a cloud server.

Figure 1 demonstrates the architecture of the proposed OLMDRODTS technique to schedules multiple heterogeneous tasks into resource-efficient virtual machines in the dynamic cloud environment. In the cloud, multiple users' send their tasks to a cloud server. After receiving the request from the cloud, the task scheduler in the cloud uses the oppositional learnt Multi-objective dragonfly optimization to find the virtual machine based on CPU, memory, bandwidth, and energy to schedule the tasks with higher efficiency and lesser overhead. The process of the OLMDRODTS technique is described in Section 2.2.

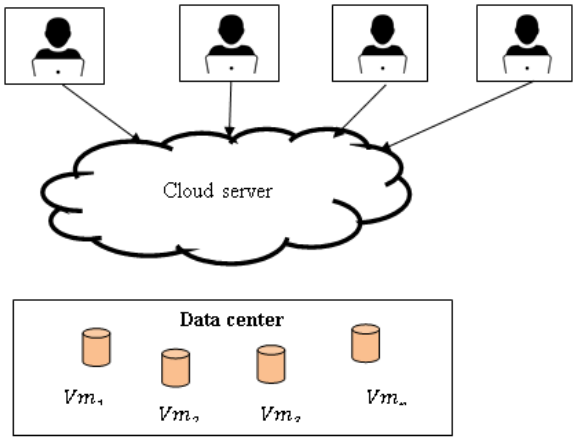

Fig. 1. Architecture of proposed OLMDRODTS technique.

2.2. *Oppositional Learnt Multi-objective Dragonfly Optimization*

The proposed OLMDRODTS technique performs dynamic task scheduling using Oppositional learnt Multi-objective dragonfly optimization. The cloud server consists of a set of virtual machines for both computational and storage facilities. The Multi-objective Oppositional Learnt Dragonfly Optimization is the meta-heuristic technique employed to find an precisely good solution in the optimization problem. Multi-objective represents the proposed optimization technique algorithm solves the multiple objective functions such as CPU, memory, bandwidth, and energy. On the contrary to existing optimization, the proposed technique uses oppositional based learning concept to achieve the global best solution among the population with minimum time consumption. Besides, the oppositional based learning optimization algorithm increases the convergence speed, flexibility, error tolerance, and higher accuracy.

The behavior of the dragonfly is moving and seeking its food source. Here the dragonfly is related to the number of virtual machines and the food source is related to

multi-objective functions i.e. CPU, memory, bandwidth, and energy. The proposed optimization algorithm worked based on the population (called a swarm). Initialize the population of the dragonfly (i.e. virtual machines) and are moved around in the search space.

Initialize the population of the virtual machines in the search space as in Equation 1.

$$Q = \{Vm_1, Vm_2, \dots, Vm_b\} \tag{1}$$

Where 'Q' indicates a current population of the virtual machines$Vm_1, Vm_2, \dots, Vm_b$. By applying the opposition based learning concept, the proposed technique generates the opposite population with the current population to achieve a global best solution. The opposition population generation is given in Equation 2.

$$Q' = m_i + n_i - Q \tag{2}$$

Where, Q'denotes an opposite population generation based on the current population 'Q', m_i and $n_i$indicates a minimum and maximum value of the dimensions in the current population 'Q'. Therefore, the current and the opposite population are generated in the search space. After the population generation, the multiple objective functions are computed for each dragonfly (i.e. virtual machine) in the current as well as the opposite population. The multiple objective functions such as availability of CPU, memory, bandwidth, and energy are calculated. Initially, the CPU is measured as the amount of time consumed by the virtual machine to complete a certain task. Therefore, the availability of CPU is estimated as given in Equation 3,

$$CPU_{AVL} = [CPU_T] - [CPU_{cd}] \tag{3}$$

From (3), CPU_{AVL} indicates the CPU availability of the virtual machine, CPU_T denotes a total CPU time of Vm, CPU_{cd} represents as consumed time of Vm to complete the particular task. Then the memory availability of the virtual machine is measured as in Equation 4.

$$M_{avl} = [M_T] - [M_{ut}] \tag{4}$$

Where, M_{avl}represents the memory availability of the virtual machine, $M_T$represents a total memory of a virtual machine,M_{ut} indicates a utilized memory space of a virtual machine. The bandwidth availability of the 'Vm' is calculated as given in Equation 5,

$$B_{avl} = [B_T] - [B_{ut}] \tag{5}$$

Where, B_{avl}indicates an available bandwidth of Vm, $B_T$indicates a total bandwidth,B_{ut}represents an amount of bandwidth consumed by the Vm. At last, the energy availability of the virtual machine is calculated as given in Equation 6,

$$\mathrm{E_{AVL}} = [\mathrm{E_T}] - [\mathrm{E_C}] \tag{6}$$

From (6), E_{AVL} indicates residual energy of the virtual machine, E_T indicates a total energy, E_C denotes consumed energy. Based on the above-calculated resource availability, the fitness is measured as given in Equation 7.

$$\varphi_F = \arg\max\{(CPU_{AVL})\&\&(M_{AVL})\&\&(Band_{AVL})\&\&\ (E_{AVL})\} \tag{7}$$

Where, φ_F denotes a fitness function, arg max({}indicates an argument of maximum function. After the fitness estimation, the current and opposite populations are combined into one and rank the dragonflies according to their fitness value. The virtual machine (Vm) has maximum resource availability is ranked first than the others as per Equation 8.

$$Vm = \varphi_F(Vm_1) > \varphi_F(Vm_2) > \ ... > \varphi_F(Vm_b) \tag{8}$$

Finally, 'n' best virtual machines are selected from the combination to find the global best solution. In order to find the global best dragonfly in the search space, four different processes are employed such as separation, alignment, cohesion, and attraction towards the food source based on their fitness. Initially, the separation process is executed to find the current position of the dragonfly and its neighboring position using Equation 9.

$$\alpha = -\sum_{j=1}^{m}\left(p\ (t) - p_j\ (t)\right) \tag{9}$$

From (9), 'α' indicates a separation process, $p\ (t)$ indicates a current position of a virtual machine at a time 't', $p_j\ (t)$ indicates a position of the neighboring virtual machine at a time 't', 'm' indicates a number of neighboring dragonflies. Secondly, the alignment process is carried out by matching the movement velocity of dragonflies and their neighborhood based on Equation 10

$$\beta = \sum_{j=1}^{m}\frac{v_j\ (t)}{m} \tag{10}$$

From (10),βindicates an alignment, $v_j\ (t)$represents a velocity of 'neighboring dragonflies, $'m$' indicates a number of neighboring dragonflies. The third process is cohesion which is thetendency of dragonflies moving towards the middle of the mass of their neighborhood as per Equation 11.

$$\gamma = \sum_{j=1}^{m}\frac{p_j(t) - p(t)}{m} \tag{11}$$

From (11), 'γ'denotes a cohesion of the dragonfly,$p_j\ (t)$be the position of the 'neighboring dragonfly, $p(t)$represents a position of a current dragonfly, $m$denotes the number of neighborhoods. Finally, the attraction towards food source is estimated as given in Equation 12,

$$\omega = p_F(t) - p(t) \tag{12}$$

Where, ωindicates an attraction function, $p_F(t)$indicates the position of the food source, $p(t)$indicates the current position of a dragonfly. Finally, the position of the dragonfly gets updated to find the global best solution based on Equation 13.

$$p_{t+1} = p(t) + \nabla p_{t+1} \tag{13}$$

From (13),p_{t+1}stand for the updated position of the dragonfly, $p(t)$is the dragonfly current position, ∇p_{t+1}indicates the step vector used to find the movement direction of the dragonfly as in Equation 14.

$$\nabla p_{t+1} = \{b_1\alpha + b_2\beta + b_3\gamma + F\omega\ \} + \ W * p(t) \tag{14}$$

Where,$b_1$designates a weight of separation (α), $b_2$indicates a weight of alignment β, $b_3$denotes a weight of cohesion γ, 'F'represents a food vector, ωindicates an attraction towards a food source, 'W'indicates an inertia weight used to controls the convergence behavior of optimization, $p(t)$ represents a current position of the dragonfly at time 't'. As a result, the global best dragonfly is identified from the updated position of the dragonfly. After identifying the global best virtual machine, the scheduler assigns the incoming user-requested task with higher efficiency. The algorithmic process of the proposed OLMDRODTS technique is described as given below,

\\ Algorithm 1: Oppositional learnt Multi-objective dragonfly Resource Optimized Dynamic Task Scheduling
Input: users '$U = \{u_1, u_2, \dots, u_m\}$; Requestedtasks $T = \{T_1, T_2, \dots, T_n\}$', virtual machines '$VM = \{Vm_1, Vm_2, \dots, Vm_b\}$', task scheduler '$TS$ '
Output: Improve the task scheduling efficiency
Begin
Step 1: Collect the number of requestedtasks '$T = \{T_1, T_2, \dots, T_n\}$ from users '$u_1, u_2, \dots, u_m$'
Step 2: for each 'T'
Step 3: Initialize the virtual machine populations of$Q = \{Vm_1, Vm_2, \dots, Vm_b\}$
Step 4: Initialize the opposite population of dragonflyQ'
Step 5: For each 'Vm' in Q and Q'
Step 6: Compute multiple objective functions M_{AVL}, B_{AVL},CPU_{AVL},E_{AVL}
Step 7: Measure the fitness '$\varphi_F{}'$
Step 8: End for
Step 9 Combines two populations
Step 10: Rank the virtual machines$Vm_1, Vm_2, \dots, Vm_b$
Step 11: Select local optimum'
Step 12:Calculate $\alpha, \beta, \gamma, \omega$
Step 13:If ($\varphi_{F_i} > \varphi_{F_j}$)then
Step 14: Update the position of the dragonfly
Step 15: End if
Step 16: If (max_ iter is reached) then
Step 17: Obtain global best dragonfly
Step 18: else

Step 19: Go to step 7

Step 20:End if

Step 21: Task Scheduler allocates the tasks to the optimal virtual machine

Step 22: End for

End

Algorithm 1 expresses the step by step process of Dynamic Task Scheduling using Oppositional learnt Multi-objective dragonfly optimization. At first, initializes the populations of the current and opposite populations of virtual machines. The fitness is calculated for each virtual machine in the current and opposite population-based on multiple objective functions. After that, two populations are combined and rank the dragonfly based on their fitness value. Finally, a local optimum solution is selected for further processing. Based on the fitness estimation, four different processes namely separation, alignment, cohesion, and attraction towards the food source are estimated. Finally, the position of the dragonfly gets updated and finds an optimal solution. Then the scheduler assigns the incoming tasks into the resource optimized virtual machine with higher efficiency.

3. EXPERIMENTAL SETUP

In this section, experimental evaluation of the proposed OLMDRODTS technique and existing CSSA [1] and HIGA [2] are implemented in the JAVA platform using CloudSim simulation. In order to conduct the experiments, the Personal Cloud datasets are taken from http://cloudspaces.eu/results/datasets. This dataset includes17 attributes (i.e. columns) and it includes 66245 instances. The main aim of this dataset is to perform load and transfer tests. 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 the 17 columns, two columns are not used such as time zone and capped. With this available informations, the user requested tasks are scheduled to the resource-efficient virtual machines in the cloud data center.

4. PERFORMANCE RESULTS AND DISCUSSION

In this section, the experimental results of the OLMDRODTS technique and two existing methods namely CSSA [1] and HIGA [2] are discussed with various performance metrics such as task scheduling efficiency, false-positive rate, computation overhead, and memory consumption with respect to the number of user tasks. The performance of proposed and conventional optimization techniques are discussed using a table and graphical illustration in next section.

4.1. *Task Scheduling Efficiency*

Task scheduling efficiency is measured as the ratio of the number of tasks that are correctly scheduled to the optimized virtual machines to the total number of tasks. The task scheduling efficiency is measured as in Equation 15.

$$TSE = \left[\frac{Number\ of\ tasks\ are\ correctly\ scheduled}{n}\right] * 100 \quad (15)$$

From (15), TSE represents a task scheduling efficiency,'n' indicates a total number of user-requested tasks. The overall task scheduling efficiency is measured in terms of percentage (%).

Table 1. Task scheduling efficiency

Number of User-requested Tasks	Task Scheduling Efficiency (%)		
	CSSA	HIGA	OLMDRODTS
30	87	83	90
60	88	85	92
90	87	84	92
120	86	82	90
150	85	83	89
180	86	84	90
210	85	82	89
240	87	83	92
270	88	85	93
300	86	83	91

Table 1 summarizes the performance analysis of task scheduling efficiency using different methods namely the OLMDRODTS technique and two existing methods namely CSSA [1] and HIGA [2]. The observed results indicate that the OLMDRODTS technique provides superior performance in terms of achieving higher efficiency than the other methods. This is proved through the statistical analysis. In the first run, 30 tasks are considered for conducting the experiments. By applying the OLMDRODTS technique, 27 tasks are correctly scheduled to the resource optimized virtual machine and their efficiency is90%. Whereas 26 and 25 tasks are correctly scheduled using CSSA [1] and HIGA [2] and the efficiency are 87% and 83% respectively. For each method, ten different results are obtained and the results are compared. The comparison results indicate that the task scheduling efficiency of the OLMDRODTS technique is increased by 5% when compared to [1] and 9% when

compared to [2]. The graphical representation of the OLMDRODTS technique is illustrated in figure 2.

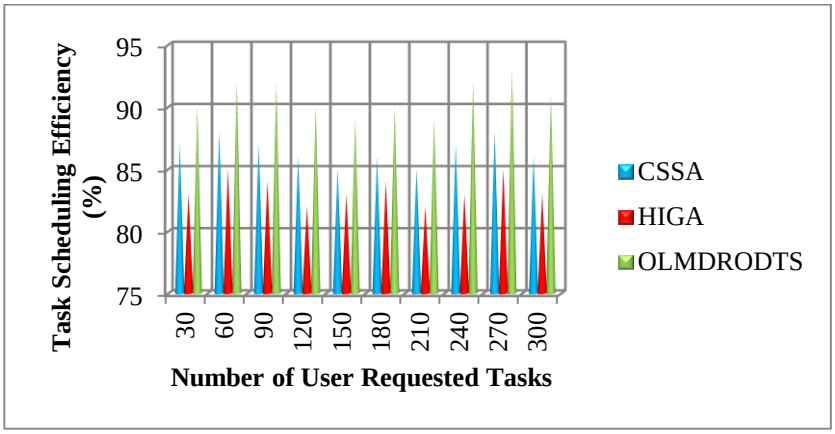

Fig. 2. Graphical Representation of Task Scheduling Efficiency

Figure 2 illustrates the task scheduling efficiency with respect to 300 user-requested tasks. These tasks dynamically arrived at the cloud server at different time. From the figure, the efficiency of the OLMDRODTS technique and existing CSSA [1] and HIGA [2] are represented by three different colors of cones such as green, blue, and red. Among the three methods, the OLMDRODTS technique outperforms well for achieving higher efficiency. This is because of the reason that the OLMDRODTS technique uses the multi-objective optimization technique for finding the optimal virtual machine. The multi-objective functions are CPU, bandwidth availability, memory availability, and energy. The task scheduler in the cloud server finds the global best solution among the population through fitness. The virtual machine with maximum availability of the resources is selected as the global best for executing a certain task. The scheduler assigns the tasks to a particular virtual machine with higher efficiency.

4.2.*False-Positive Rate*

It is measured as the ratio of a number of user-requested tasks that are in accurately scheduled to the total number of tasks taken as input. The formula for calculating the false-positive rate is expressed as in Equation 16,

$$FPR = \left[\frac{Number\ of\ tasks\ are\ incorrectly\ scheduled}{n}\right] * 100 \tag{16}$$

From (16),FPRrepresents the false-positive rate, 'n' represents the total number of user-requested tasks. The false-positive rate is measured in percentage (%).

Table 2: False-Positive Rate

Number of User-Requested Tasks	False-Positive Rate (%)		
	CSSA	HIGA	OLMDRODTS
30	13	17	10
60	12	15	8
90	13	16	8
120	14	17	10
150	15	18	11
180	14	16	10
210	15	18	11
240	13	17	8
270	12	15	7
300	14	17	9

Table 2 portrays the performance results of false-positive rate according to the number of user requests 30 to 300. The result indicates that the OLMDRODTS technique minimizes the incorrect task scheduling in a cloud environment than the other existing methods. By considering 30user-requested tasks for experimentation in the first run, 3 tasks are incorrectly scheduled and the false positive rate is 10% using the OLMDRODTS technique. Followed by, the 4 and 5user-requested tasks are incorrectly scheduled using CSSA [1] and HIGA [2], and data false positive rate percentages are 13% and 17% respectively. Totally, ten runs are observed for each method with a different number of user tasks. Finally, the observed result of the proposed OLMDRODTS technique is compared with the results of existing methods. The average of comparison results indicates that the false positive rate of the OLMDRODTS technique is minimized by32% and 45% when compared to CSSA [1] and HIGA [2] respectively. The performance result of the false positive rate is shown in figure 3.

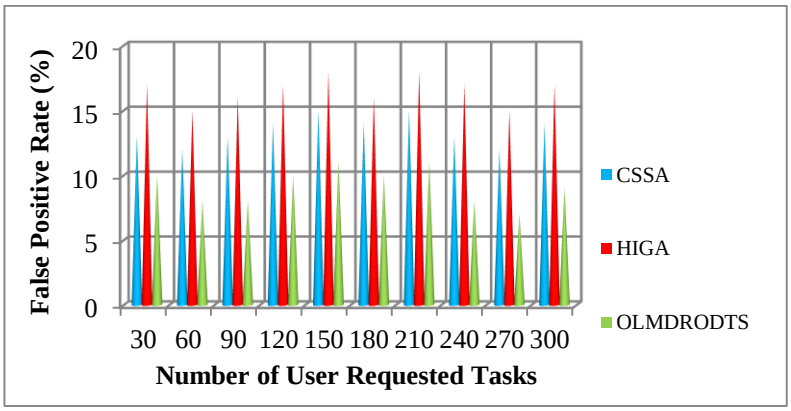

Fig. 3. Graphical Representation of the False-Positive Rate

Figure 3 given above demonstrates the comparative analysis of the false-positive rate of incorrect task scheduling in the cloud. As revealed in the chart, the ‘horizontal axis

refers to the number of user-requested tasks and the vertical axis refers to the results of the false-positive rate. From the graphical plot, it is inferred that the false-positive rate of the OLMDRODTS technique is minimized than the other methods. This is because the OLMDRODTS technique uses the oppositional learning concept to the multi-objective dragonfly optimization for generating the opposite population along with the current population to improve the fault tolerance. This process helps to find the global optimum in the search space for assigning the tasks. Then the task scheduler efficiently assigns the task to the virtual machine to minimize the false positive rate.

4.3.*Computation Overhead*

Computation overhead is measured as a time taken to schedule the given tasks into the resource optimized virtual machines. The computation overhead is measured using the following equation 17.

$$CO = n * t(schedule\ one\ task) \tag{17}$$

From (17), 'CO' indicates a computation overhead, n' indicates a total number of tasks, and 't' denotes a time taken to schedule the single task. The computation overhead is measured in milliseconds (ms).

Table 3: Computation Overhead

Number of User-Requested Tasks	Computation Overhead (ms)		
	CSSA	HIGA	OLMDRODTS
30	22	24	19
60	24	27	22
90	27	30	25
120	31	34	29
150	35	36	32
180	37	39	35
210	40	42	38
240	43	46	41
270	46	49	44
300	50	52	47

Table 3 reports the performance of computation overhead using three different methods namely the OLMDRODTS technique and two existing methods CSSA [1] and HIGA [2] with respect to the number of users requested tasks. From the tabulated results, it is inferred that the computation overhead using the OLMDRODTS technique is comparatively lesser than the [1] and [2]. Initially, the experiment is

conducted with 30 tasks, OLMDRODTS technique consumes the time of $19ms$ to schedule the tasks. Similarly, the time consumption of the existing methods are $22ms$ and $24ms$. Likewise, the other remaining nine runs are carried out with different user-requested tasks. The overall observed results indicate that the proposed OLMDRODTS technique minimizes the computation overhead by 7% and 13% when compared to existing methods. Figure 4 clearly illustrates the experimental results of computation overhead versus a number of tasks. For each method, ten varieties of results are observed. The time is found to be in the increasing trend for all the three methods while increasing the number of tasks. But the results clearly show that the computation overhead is found to be considerably minimized using the OLMDRODTS technique as compared to existing scheduling techniques. This is due to the application of oppositional learning concept and ranking to the Multi-objective dragonfly optimization. The ranking method finds the local optimum from the population. In order to perform the scheduling with more virtual machines, the technique consumes more time. But the OLMDRODTS technique uses the ranking method to sort the virtual machines based on the fitness. Finally, the global optimum solution is determined from the local optimum. This process takes a minimum time to find the resource-efficient virtual machine and it also reduces the scheduling time of large number of tasks.

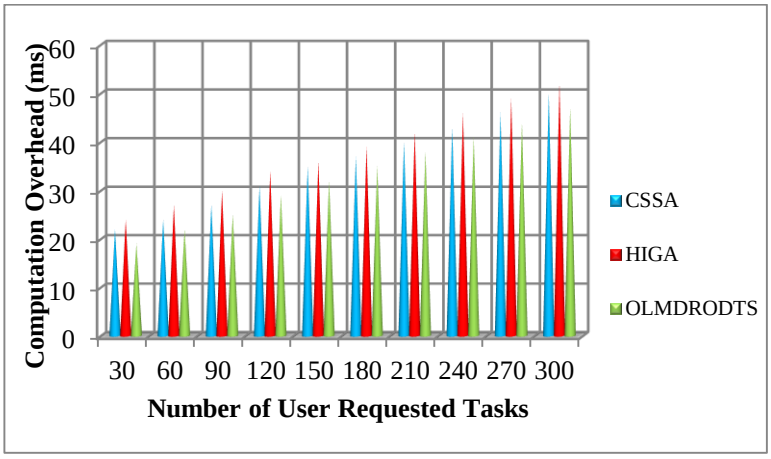

Fig. 4. Graphical Representation of Computation Overhead

4.4.*Memory Consumption*

Memory consumption is the amount of memory space taken by the virtual machine to schedule the user tasks. The overall memory consumption is estimated as given in Equation 18.

$$MC = n * M\ (SST) \tag{18}$$

Where MC represents a memory consumption, 'n' denotes a total number of user-requested tasks, '$M\ (SST)$' indicates the memory consumption to schedule the single-user tasks and. Therefore, the overall memory consumption is measured in the unit of megabytes (MB).

Table 4: Memory Consumption

Number of User-Requested Tasks	Memory Consumption (MB)		
	CSSA	HIGA	OLMDRODTS
30	20	23	17
60	23	25	21
90	26	28	24
120	29	31	26
150	32	33	30
180	34	36	32
210	36	38	34
240	38	41	36
270	40	43	38
300	42	45	40

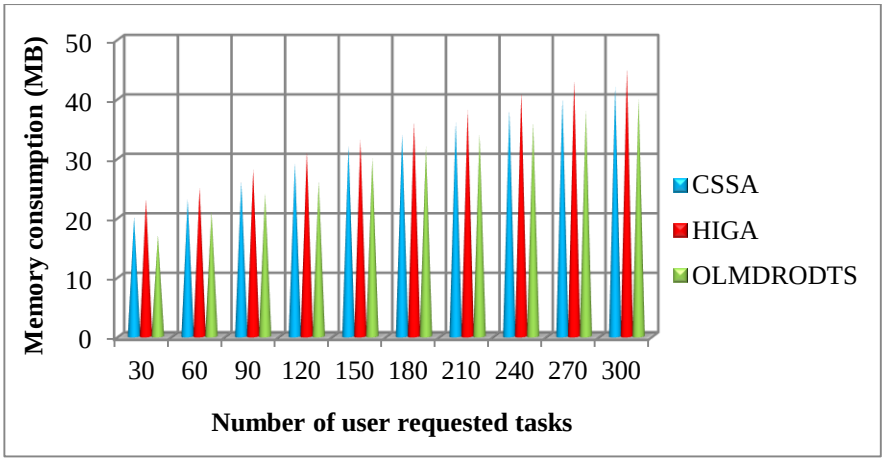

Fig. 5. Graphical Representation of Memory Consumption

Table 4 and Figure 5 given describe the performance of memory consumption using three different methods namely the OLMDRODTS technique and two existing methods namely CSSA [1] and HIGA [2] with respect to the number of tasks. Among the three methods, the OLMDRODTS technique consumed lesser memory space for scheduling the given user request to the cloud server. Initially, the experiment is conducted with 30tasks, OLMDRODTS technique consumes the $17MB$memory space during the scheduling process whereas the space consumption of the CSSA [1] and HIGA [2] are $20MB$ and $23MB$ respectively. Similarly, the remaining nine runs are carried out and the results are compared. The average of ten results indicates the

memory consumption of the OLMDRODTS technique is significantly reduced by 7% and 14% when compared to existing [1] and [2] respectively. This is due to the proposed technique discovers the resource optimized virtual machine to schedule the multiple heterogeneous user tasks.

5. CONCLUSION

In this research work, an efficient technique called the OLMDRODTS technique is introduced for energy-efficient task scheduling process and dynamic resource allocation in cloud environments. In the proposed task scheduling technique, the oppositional learning concept is applied in the Multi-objective dragonfly optimization to find the global optimum resource-efficient virtual machine based on the CPU time, memory, bandwidth, and energy. The resource optimized virtual machine is selected based on the fitness measure. Finally, a task scheduler in a cloud server assigns the incoming tasks to the resource optimized virtual machine. Then the virtual machine executes the assigned tasks with the minimum response time. To evaluate the performance of the OLMDRODTS technique and other scheduling techniques, a personal cloud dataset is implemented in the cloudsim simulator four different metrics. The statistical analysis indicates that the OLMDRODTS technique provides better performance for dynamically scheduling a large number of tasks with higher efficiency and minimum overhead as well as memory consumption than the state-of-the-art works.

REFERENCES

[1] M.S.Sanaj, P.M.Joe Prathap, “Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere”, Engineering Science and Technology, an International Journal, Volume 23, Issue 4, 2020, Pages 891-902

[2] Mohan Sharma and Ritu Garg, “HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers”, Engineering Science and Technology, an International Journal, Elsevier, Volume 23, Issue 1, 2020, Pages 211-22.

[3] J. Praveenchandar and A. Tamilarasi, “Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing”, Journal of Ambient Intelligence and Humanized Computing, Springer, 2020, Pages 1-13.

[4] Seema A. Alsaidy, Amenah D. Abbood, Mouayad A. Sahib, “Heuristic initialization of PSO task scheduling algorithm in cloud computing”, Journal of King Saud University - Computer and Information Sciences, Elsevier, 2020, Pages 1-13.

[5] Shanchen Pang, Wenhao Li, Hua He, Zhiguang Shan, Xun Wang, "An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing", IEEE Access , Volume 7, 2019, Pages 146379 – 146389.

[6] Sarah E. Shukri, Rizik Al-Sayyed, Amjad Hudaib, Seyedali Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments", Expert Systems with Applications, 2020, Pages 1-30

[7] M.S.Sanaj and P.M.Joe Prathap, "An efficient approach to the map-reduce framework and genetic algorithm based whale optimization algorithm for task scheduling in cloud computing environment", Materials Today: Proceedings, Elsevier, 2020, Pages 1-10

[8] Xuan Chen, Long Cheng, Cong Liu, Qingzhi Liu, Jinwei Liu, Ying Mao, John Murphy, "A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems", IEEE Systems Journal , Volume 14, Issue 3, 2020, Pages 3117 – 3128

[9] Xingwang Huang, Chaopeng Li, Hefeng Chen & Dong An, "Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies", Cluster Computing, Springer, Volume 23, 2020, Pages 1137–1147

[10] Reihaneh Khors and Mohammadreza Ramezanpour, "An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing", International Journal of Communication Systems, Wiley, Volume 33, Issue 9, 2020 , Pages 1-17

[11] Seyedakbar Mostafavi & Vesal Hakami, "A Stochastic Approximation Approach for Foresighted Task Scheduling in Cloud Computing", Wireless Personal Communications, Springer, Volume 114, 2020, Pages 901-925.

[12] Ding Ding, Xiaocong Fan, Yihuan Zhao, Kaixuan Kang, Qian Yin, Jing Zeng,"Q-learning based dynamic task scheduling for energy-efficient cloud computing", Future Generation Computer Systems, Elsevier, Volume 108, 2020,Pages 361-371.

[13] Harvinder Singh, Sanjay Tyagi, Pardeep Kumar, "Crow–penguin optimizer formultiobjective task scheduling strategy in cloud computing", International Journal of Communication Systems, Wiley, Volume 33, Issue 14, 2020, Pages 1-18.

[14] Belal Ali Al-Maytami, Pingzhi Fan, Abir Hussain, Thar Baker, Panos Liatsis, "A Task Scheduling Algorithm With Improved Makespan Based on Prediction of Tasks Computation Time algorithm for Cloud Computing", IEEE Access, Volume 7, 2019, Pages 160916 – 160926.

[15] T. Prem Jacob & K. Pradeep, "A Multi-objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle Swarm Optimization", Wireless Personal Communications, Springer, Volume 109, 2019, Pages 315-331.

[16] Mitali Bansal and Sanjay Kumar Malik, "A multi-faceted optimization scheduling framework based on the particle swarm optimization algorithm in cloud computing", Sustainable Computing: Informatics and Systems, Elsevier, Volume 28, 2020, pages 1-8.

[17] Liqiong Chen, Kun Guo, Guoqing Fan, Can Wang, Shilong Song, "ResourceConstrained Profit Optimization Method for Task Scheduling in Edge Cloud",IEEE Access Volume 8, 2020, Pages 118638 – 118652.

[18] Yi Gu and Chandu Budati, "Energy-aware workflow scheduling and optimizationin clouds using bat algorithm", Future Generation Computer Systems, Volume 113, 2020, Pages 106–112.

[19] Shridhar G. Domanal, Ram Mohana Reddy Guddeti, Rajkumar Buyya, "A HybridBio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment", IEEE Transactions on Services Computing, Volume 13, Issue 1, 2020, Pages 3 – 15.

[20] Tingting Dong, Fei Xue, Chuangbai Xiao, Juntao Li, "Task scheduling based ondeep reinforcement learning in a cloud manufacturing environment", Concurrency and Computation: Practice and Experience, Wiley, Volume 32, Issue 11, 2020, Pages 1-12.