Fuzzy Systems and Soft Computing
ISSN : 1819-4362

GENETIC DOVE SWARM OPTIMIZATION ALGORITHM-BASED TASK SCHEDULING MODEL FOR SOLVING THE EXECUTION DELAY PROBLEM IN IAAS CLOUDS

D.THANGAMANI, Research Scholar, Department of Computer Science, Mother Teresa Women's University Kodaikanal.
Dr.K.KUNGUMARAJ,Assistant Professor ,PG Department of Computer Science, Arulmigu Palaniandavar Arts College for Women,Chinnakalayamputhur,Palani.

Abstract –
Task scheduling (TS) is a prominent process in cloud systems. The inappropriate selection of task scheduling procedures degrades the performance of cloud systems. Different optimization techniques are used for resolving task scheduling problems. However, these techniques consider bandwidth and execution parameters for allocating resources. These parameters did not handle the execution delay problem related to task submission duration. Therefore, this paper proposes the Genetic Dove Swarm Optimization (GDSO) algorithm, a hybrid technique for solving the multi-objective problem of TS in IaaS cloud services. The GDSO technique is developed by integrating Genetic Algorithm (GA) and Dove Swarm Optimization (DSO) technique. This method provides maximum quality-of-service (QoS) and curtails execution delay problems. Here, problems occurred in TS is taken as NP-complete problem, and the proposed GDSO resolve this problem based on multiple parameters like execution time, consumption of energy, response time and the total expected cost. As a result, GDSO allocates the tasks effectively without execution delay. The performance of GDSO is evaluated utilizing the CloudSim tool, and it observed that GDSO is more suitable for scheduling tasks in cloud systems than the compared methods, with better energy consumption, execution time, execution cost and response time.
Keywords: Task Scheduling, Genetic Dove Swarm Optimization, Genetic Algorithm, multi-objective, execution delay problem, CloudSim.

1. INTRODUCTION

In the distributed computing field, cloud computing is considered an emerging technology. Applications like data storage, management and processing are done with cloud computing. Cloud data is spread across multiple servers linked together via networked resources and is accessed using virtual machines [1]. Cloud computing offers many services, which include Platform as a Service (PaaS), Software as a Service (SaaS) and Infrastructure as a Service (IaaS) [2]. PaaS is deployed in cities or particular organizations and comes under the private cloud category. In contrast, SaaS permits service provision within the country and is considered a public cloud. The combination of public and private clouds is IaaS which has the benefit of services of public and private clouds. The increased usage of cloud services resulted in reduced throughput. This diminishes the efficiency of the cloud system. In addition, a practical TS algorithm must be selected to ensure the quality of service of the cloud system [3]. TS is performed to improve the resource utilization ability, which maps the task to a particular VM.

The task scheduling must reduce the system's makespan to provide better service quality. Therefore discovering the optimal solutions for task scheduling problems is recognized as the NP-Complete [4]. The near-optimal solution for the NP-Complete problem is obtained by exploring the large solution search space using metaheuristic algorithms. These large-scale problems are recently solved utilizing heuristic as well as metaheuristic algorithms. In general, Ant Colony Optimization (ACO) [5], Artificial Bee Colony (ABC), GA, Bat algorithm and Particle Swarm Optimization (PSO) are used for task scheduling [6]. However, it has some limitations due to its high computational time and improper balancing of global and local search procedures. These algorithms are often trapped in local optima and imbalance between the search procedures resulting in premature convergence. Moreover, the heuristic algorithm not improves the solution quality for large-scale problems. Metaheuristic algorithms solve complex optimization-related problems but are often trapped in local optima problems. Hence it requires more memory and reduces the convergence rate [7]. This limitation affects the performance of task scheduling. Therefore, the drawbacks of optimization techniques are resolved by combining the merits of both heuristic and metaheuristic approaches.

In this paper, GA is merged with Dove Swarm Optimization (DSO) Algorithm to develop a hybrid algorithm named Genetic Dove Swarm Optimization (GDSO) for solving the TS problem. This method considers the TS process as a multi-objective problem and solves this problem through an effective optimization approach. This method uses the foraging behaviour of the dove swarms from the DSO algorithm to solve the multi-objective TS problem. GA enhances the DSO performance to obtain the optimal solution with reduced delay. This hybrid algorithm reduces the cost of execution, response time, energy consumption, execution time, bandwidth utilization and CPU utilization. The main objectives of this paper are to formulate the Multi-objective problem related to task scheduling so that the performance of the task scheduling process with reduced execution delay problem.

This paper is arranged as follows: The points noted in existing work are presented in Section 2. The proposed GDSO technique is explained in Section 3. The performance achieved by GDSO and existing methods in task scheduling is discussed in Section 4, and conclusions are presented in Section 5.

2. LITERATURE REVIEW

Many optimization algorithms have been utilized in recent years for the problem of TS based on multiple criteria. Some of the most recent methods are discussed in this section. Gawali and Shinde [8] integrated improved Bandwidth Aware Divisible Scheduling (BATS), Longest Expected Processing Time Pre-emption (LEPT), Modified Analytic Hierarchy Process (MAHP) and divide-and-conquer technique for resource allocation and task scheduling in cloud systems. Initially, MAPH is implemented to assign a rank to each task. Then the task allocation procedure is carried out using the improved BATS technique. Next, LEPT is implemented to prevent tasks that are resource intensive. Further, Divide and Conquer procedure provides an optimal solution to enhance the model's performance. This model is evaluated using Epigenomics scientific tasks, and Cybershake seismogram synthesis datasets and the performance is measured using the utility of bandwidth, CPU and memory. The performance of this model is compared with BATS and the improved differential evolution algorithm (IDEA), and it observed that improved BATS gained better performance. However, this model has a low convergence speed. Kumar and Kousalya [9] deployed Crow Search Algorithm (CrSA) to schedule the tasks in cloud systems. In this approach, the makespan of task scheduling is reduced using CrSA. This model is implemented using the CloudSim tool, and observed that it reduces the makespan more than prevailing models. However, this model considers only makespan in task scheduling. Bezdan et al. [10] used Exploration Enhanced- Flower Pollination Algorithm (EE-FPA) for TS in cloud. In this approach, the makespan of task scheduling is minimized using EEFPA. The effectiveness of this model is tested on the CloudSim tool and noted that it had a reduced makespan of 6.67% compared to FPA, 4.11% for Performance Budget –ACO (PBACO), 53.02 for ACO, 29.29% for min-min and 63.92% for First Come First Serve (FCFS) allocation method while analyzing with 600 tasks. However, this model not considers other factors affecting the task scheduling process of the cloud system. Rjoub et al. [11] developed a model utilizing deep reinforcement learning combined with LSTM (DRL-LSTM) for task scheduling. In this approach, resource utilization on TS is reduced. This model was evaluated on the Google cluster dataset. The DRL-LSTM model reduced the utilization of RAM cost by 72%, which is better than other methods. However, this model has increased computational overhead.

Velliangiri et al. [12] enhanced the TS behaviour using Hybrid Electro Search with GA (HESGA). In this approach, the best local solution is obtained using GA, whereas the global best solution is achieved utilizing Electro Search (ES) technique. This model is experimented with the Cloudsim tool and noted

that it has better makes span, response time and cost than Hybrid PSO-GA, ES, ACO and GA. However, a lesser number of parameters is taken to analyze the model. Elaziz and Attiya [13] referred to the improved Henry Gas Solubility Optimization (HGSO) technique for efficient TS in the cloud. This model employs WOA and opposition-based learning approaches to enhance HGSO's performance in task scheduling. This modification increases the converging speed. This model is examined on real NASA iPSC and synthetic datasets, achieving a better makespan. However, this model has high complexity.
Abualigah and Diabat [14] modified the antlion optimization algorithm (AOA) for effective task scheduling. In this approach, AOA is improved using elite-based differential evolution to diminish the makespan and utility of resources. The performance of improved AOA is tested on real and synthetic datasets. On experimenting, modified AOA has faster convergence than other methods, which shows its ability to work on significant scheduling problems. The modified AOA approach attained a 16.55 % improved ratio on 2000 tasks for the HPC2N dataset and 17.01% for the NASA Ames dataset with the same number of tasks. However, this method used few parameters for performance evaluation. Gupta et al. [15] resolved the task scheduling issues by performing a few modifications in Heterogeneous Earliest Finish Time (HEFT) technique. In this approach, the ranks of each task and idle slots are computed to enhance the task-scheduling process. Furthermore, it optimally assigns the resources to the cloud environment. This model's effectiveness is tested on 100 diverse problems with a size of 80. The performance is measured as schedule length, and it is noted that the updated HEFT algorithm performs better than others. However, the complexity of this method is not reduced.
Bal et al. [16] suggested Cat Swarm Optimization (CSO) and Group Optimization-based Deep Neural Network (GO-DNN) methods for task scheduling and resource allocation in the cloud environment. This method increases the throughput utilizing CSO's optimal scheduling approach and reduces the total makespan. These techniques are simulated utilizing a cloudlet simulator. The effectiveness is measured as energy consumption, resource utilization and response time and observed that the integration of CSO and GO-DNN performs better than other models. However, the effectiveness of this model is not tested on real-time data. Abdullahi et al. [17] developed a constrained multi-objective Adaptive Benefit Factors Based Symbiotic Organisms Search (ABFSOS) methodology for TS. This approach increases convergence speed by balancing the global and local search procedures. This model is evaluated utilizing synthetic and standard workload datasets on the Cloudsim tool. The performance of ABFSOS is compared with existing EMS-C and ECMSMOO approaches and attained the performance improvement of 17.02-47.73% for EMS-C and 19.98-52.18% for ECMSMOO. However, this model has high computational complexity. Amer et al. [18] enhanced the performance of cloud resources by task scheduling utilizing the updated Harris Hawks Optimization (HHO) technique. This method solves the multi-objective scheduling problem using the HHO method. Furthermore, HHO uses an opposition-based learning approach to improve its exploration quality. This model is evaluated on the real-time dataset, and its effectiveness is contrasted against existing models with the performance metrics such as balancing degree, resource utilization, throughput, schedule length and execution cost. While analyzing the result, modified HHO performs better than other models.
Bezdan et al. [19] propose a hybrid bat algorithm (BA) for task scheduling. At first, the bee search algorithm is implemented to balance the exploration and exploitation process that BA does not perform. In addition, the quasi-reflection-based learning (QRBL) approach is employed to enhance searchability. Using these approaches, task scheduling effectiveness was improved regarding makes span and cost of operation. This model was evaluated on HPC2N, NASA Ames iPSC/860 and synthetic datasets with the Cloudsim tool. However, this model is hard to implement. Manikandan et al. [20] integrated Random Double Whale Optimization Algorithm (RD-WOA) and Bee Algorithm (BA) for effectively scheduling the tasks. In this approach, the multi-objective task scheduling problem is solved using RDWHOA. Then, mutation operators of BA are implemented on RDWHOA to achieve the optimal solution for TS. This model is executed on the Cloudsim tool, and noted that it reduces the time taken for execution. Rajakumari

et al. [21] improved task scheduling using Fuzzy Hybrid-Particle Swarm Parallel ACO (FH-PSPACO) technique. In this approach, Dynamic Weighted Round-Robin (DWRR) procedure is implemented as the first step for task scheduling. The second step solves the execution delay problem in DWRR utilizing the hybrid PSPACO method. Finally, task scheduling is implemented with the assistance of the fuzzy logic system. This hybrid model is executed using the Cloudsim tool. While experimenting, FH-PSPACO outperformed the standard DWRR and PSPACO models.

Abualigah and Alkhrabsheh [22] recommended the TS method based on a hybrid Multi-Verse Optimizer with GA (MVO-GA). In this approach, the performance of task transfer is improved by utilizing the MVO-GA method. This model is implemented on MATLAB tool and evaluated using different datasets. The experimental outcome reveals that MVO-GA performs better on task scheduling than prevailing approaches. However, the effectiveness of this model is not assessed on a large dataset. Imene et al. [23] used the third-generation Non-dominated Sorting GA (NSGA-III) method for task scheduling. This approach resolved the multi-objective problem of task scheduling utilizing the NSGA-III procedure. Different parameters are taken for the performance evaluation of NSGA-III, and it is compared against its previous version NSGA-II. NSGA-III decreased the power consumption, runtime and operational cost. Nabi et al. [24] introduced an optimal TS approach based on Adaptive PSO (APSO) technique. The APSO technique achieves optimal scheduling in this approach, balancing local and global searches. This model is evaluated on the Cloudsim tool and observed that it attains the performance improvement of 10% on makespan, 12% on throughput and 60% on resource utilization. However, the response time of this system is not significant.

Mahmoud et al. [25] employed Decision Tree (DT) algorithm for TS in the cloud environment. This approach minimizes the makespan and maximizes the utility of resources using the DT method. The benchmarks taken for evaluation are CyberShake_30, Montage_25, Epigenomics_24 and SIPHT_30 workflows. DT has improved resource utilization by 6.81%, 4.69% and 8.27%. However, the response time of this system is not significant. Pradeep et al. [26] fused Cuckoo Search Algorithm (CSA) and WOA and named CWOA to enhance the TS process in cloud computing. This approach reduces energy usage and enhances the QoS of the cloud system. CWOA has improved the makespan, which is 5.62%, 2.27% and 4.36%; memory utilization is 19.08%, 16.75% and 19.34%. Kruekaew and Kimpan [27] proposed an effective TS approach utilizing the ABC and Q-learning algorithms. In this approach, the speed of the ABC algorithm is increased using the Q-learning algorithm. This model's performance is assessed using Google Cloud Jobs, Random and synthetic workload datasets. While experimenting, ABC with Q learning in multi-objective optimization performs better than existing approaches. However, this model lacks generalizability.

From the review, it is inferred that optimization techniques are mostly preferred to resolve TS problems in the cloud. However, many limitations exist in prevailing methods to effectively schedule the tasks like execution delay, increased complexity and reduced convergence speed. In this review, particular models solved the TS problem as a multi-objective problem. Still, its effectiveness is not explained with all the important parameters to predict task scheduling performance.

3. METHODS

3.1. Modelling TS problem

The TS problem is considered a multi-objective problem, and it is hard to obtain the optimal solution for such problems. This problem is also accounted as an NP-complete problem, and it is complex to find the best solution within the polynomial time. This problem is solved using the GDSO methodology. The benefits of GA and DSO are fused to form GDSO. The GA solves multi-objective problems effectively but may be stuck in local optima and not provide reliable, optimal solutions. The DSO technique is highlighted for its effectiveness in solving the optimization problem and better time efficiency. Therefore,

the benefits of GA and DSO are integrated to resolve the multi-objective task scheduling problem. Using the GDSO approach, execution delay in TS is reduced, and it improves the cloud system's QoS.
The multi-objective problem of TS is formulated as,

$$f(x) = \sum_k a_k f(x_k);\ 0 < k \le n \qquad (1)$$

Where, a_k denotes the assigned weight, $f(x_k)$ indicates the individual fitness function at $0 < k \le n$. The fitness function should generate a minimum value to achieve an effective solution. Therefore, the proposed GDSO model's multi-objective fitness function for TS is modeled as an NP-complete problem. It is expressed as,

$$F(x) = (a_1 \times TEC) + (a_2 \times ET) + (a_3 \times TS_{Res}) + \left(a_4 \times E_{ij}\right) \qquad (2)$$

Here, a_1, a_2, a_3, a_4 are the weight values and $a_1 + a_2 + a_3 + a_4 = 1$.
Total expected cost (TEC): It depends on the cost of resources taken to process and transfer the files. The TEC is computed as,

$$TEC = P_i \times P_c + \left\{\sum_{f \in FIN_i} size(f) + \sum_{f \in FOUT_i} size(f)\right\} \times PTPB \qquad (3)$$

Where, P_c represents the processing cost, P_i denotes the task, f specifies the files, and PTPB denotes processing time per byte.
Execution time (EC): The time required to complete a specific task is known as EC.

$$ET = \sum_{j \in M} \frac{ET_{i.j}}{M} \qquad (4)$$

Here, $ET_{i,j}$ represents the predicted time of the completed task that is v_i, i and j denote the tasks.
Response time: The time taken between the execution of tasks and task completion is defined as response time. In general, it is defined as the elapsed time between the request and the execution of the request. It is expressed as follows,

$$TS_{Res} = TS_{CT} - TS_{AT} \qquad (5)$$

Here, TS_{CT} represent the task completion time and TS_{AT} indicates the time taken to execute the tasks.
Energy consumption: It is the consumed energy by cloud servers while allocating resources. The consumed energy for i-th task on j-th VM is E_{ij} and it is expressed as,

$$E_{ij} = P(U_{ij}) \times CT_{ij} \qquad (6)$$

Where $P(U) = \left(P(U_1) + \left(\frac{P(U_2)-P(U_1)}{10} \times (U - \frac{U_1}{10} \times 100\right)\right)$

$$CT_{ij} = \sum_{i=1}^{N} Ft_i - St_i$$

$$U_{ij} = \frac{U_{iref} \times CS_{ref}}{CS_j}$$

Here, CT_{ij} denotes the completion time, U_{ij} represents the resource consumption, $P(U_{ij})$ specifies the depleted power, $P(U)$denotes the depleted power at U, Ft_i indicates the time taken to complete the task of j-th VM, St_i denotes the j-th VM's starting time, U_{iref} represents i-th task's power consumption on reference VM, CS_{ref} indicates the clock speed of the reference VM, CS_j indicates the clock speed of j-th VM.

3.2. Genetic Dove Swarm Optimization for task scheduling

The GDSO technique is proposed to resolve the multi-objective problem of the TS process. This approach uses the advantages of GA and DSO scheduling techniques, which limit local optima and increase convergence speed. In this approach, the global search ability of doves in DSO is enhanced using the genetic algorithm operators. The utilized operators of GA are crossover and mutation, which balances the searching ability of the dove. The DGSO technique uses the foraging behaviour of the dove in task scheduling. Initially, the parameters such as the initial position of the Dove, Number of doves, epoch and satiety degree are given as input to initiate the GDSO algorithm. The fitness value for each dove is computed.

Further, the ranks are allocated for the doves with respect to fitness value and arranged in descending order. Then the genetic operation is performed for the ranked $2N$ individuals in the flock. This genetic operation creates a new set of solutions, and fitness is computed for this solution. Next, the dove's position near the fitness value is located. The satiety level of each dove is computed to find the dove with the best satiety and is considered the dove that offers the best solution. Finally, the dove having the best solution is selected, and the locations of other doves are updated based on the selected dove. Once the best solution is discovered, the termination condition is used; otherwise, the iterations are performed continuously until obtaining the best solution. The functional flow of the proposed GDSO is illustrated in figure 1.

According to the rank, which is assigned based on the doves' fitness function, the best-ranked doves are selected for performing the genetic operations. Initially, $2N$ individuals are paired, and it is crossed over to generate new offspring. In this stage, 20% mutation is carried out to create new individuals with better fitness values. After the improvement of fitness values then, the order of ranking is rearranged. Further, Doves are allocated in hierarchical order to form the group. Finally, the formed group with the dove that does not satisfy the fed initiates the foraging process, and the best location is determined by the dove having the best solution. This process is performed for maximum iterations to discover the best location where the solution is found. The best location is continuously updated to improvise the effectiveness of finding optimal solutions.

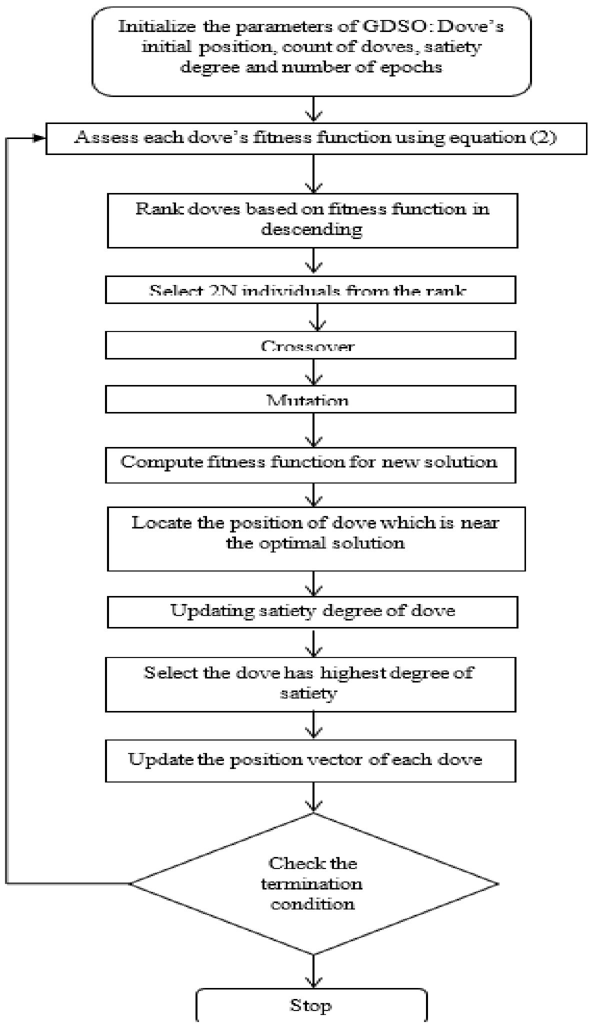
Initialize the parameters of GDSO: Dove's initial position, count of doves, satiety degree and number of epochs
Assess each dove's fitness function using equation (2)
Rank doves based on fitness function in descending
Select 2N individuals from the rank
Crossover
Mutation
Compute fitness function for new solution
Locate the position of dove which is near the optimal solution
Updating satiety degree of dove
Select the dove has highest degree of satiety
Update the position vector of each dove
Check the termination condition
Stop

Figure.1.Flow Diagram of Proposed GDSO Method

The objective function of GDSO is $f(\underline{W})$. Here $\underline{W}$ denotes the position of crumbs and amount of crumbs in that location. The location has enormous crumbs and is considered the best location.

At first, the counts of doves are decided to deploy in the search space. Assume N is the number of doves randomly dispersed in the solution space having a uniform rectangular region. Next, the epoch number is defined that is $e = 0$ and the degree of satiety S_d^e for d-th dove is fixed, where $d = 1,2 \dots\dots N$. The position vector is initialized as $\underline{W_d} \subset R^M$ for d-th dove. This initialization could be done in two ways: random and lattice-initialized methods. In this method, weight initialization is performed using two weight initialization schemes which enhance the training process. Assume the smallest hyper-rectangle, which includes the valid parameter values, and it is represented as $[l_{1,}u_1], \dots\dots [l_{M,}u_M]$ where, l_M and u_M representing the low and upbound of M dimensions in the search space. The suggested initialization procedure is to compress the hyper–rectangle with n-dimensions into a 2-dimensional plane to cover the solution space effectively.

This method, i and j are used to represent the rectangular cells from 1 to $A \times B$. The initialization steps are explained as follows. At first, cells on four corners are initialized, and their formulation is expressed in equation (7).

$$\underline{w}_{1,1} = (l_1, l_2, \dots . l_M)^T$$
$$\underline{w}_{A,B} = (u_1, u_2, \dots . u_M)^T$$
$$\underline{w}_{1,B} = \left(l_1, l_2, \dots . l_{\left\lfloor\frac{M}{2}\right\rfloor}, u_{\left\lfloor\frac{M}{2}\right\rfloor+1}, \dots . u_M\right)^T$$
$$\underline{w}_{A,1} = \left(u_1, u_2, \dots . u_{\left\lfloor\frac{M}{2}\right\rfloor}, l_{\left\lfloor\frac{M}{2}\right\rfloor+1}, \dots . l_M\right)^T \quad (7)$$

Next, cells are initialized in the four edges, and it is formulated as follows:

$$\underline{w}_{1,j} = \frac{w_{1,B}-w_{1,1}}{B-1}(j-1) + \underline{w}_{1,1} = \frac{j-1}{B-1}\underline{w}_{1,B} + \frac{B-j}{B-1}\underline{w}_{1,1} \text{ for } j = 2, \dots\dots . B-1$$
$$\underline{w}_{A,j} = \frac{w_{A,B}-w_{A,1}}{B-1}(j-1) + \underline{w}_{A,1} = \frac{j-1}{B-1}\underline{w}_{A,B} + \frac{B-j}{B-1}\underline{w}_{A,1} \text{ for } j = 2, \dots\dots . B-1$$
$$\underline{w}_{i,1} = \frac{w_{A,1}-w_{1,1}}{A-1}(i-1) + \underline{w}_{1,1} = \frac{i-1}{A-1}\underline{w}_{A,1} + \frac{A-i}{A-1}\underline{w}_{1,1} \text{ for } i = 2, \dots\dots . B-1$$
$$\underline{w}_{i,B} = \frac{w_{A,B}-w_{1,B}}{A-1}(i-1) + \underline{w}_{1,B} = \frac{i-1}{A-1}\underline{w}_{A,B} + \frac{B-i}{B-1}\underline{w}_{1,B} \text{ for } i = 2, \dots\dots . B-1 \quad (8)$$

Next, the corner neuron's weight vectors are initialized. Further, the remaining neurons are initialized from top to bottom and left to right. The learning rate's initial value is fixed as 0.1, and the reduction rate is represented as,

$$\eta(n) = \eta_0 \times \left(1 - \frac{t}{T}\right) = 0.1(1 - \frac{t}{100}) \quad (9)$$

Where η_0 representing the learning rate, and t denotes the iterative number. Next, the fitness value, $f(w_j^e)$ that is $F(x)$ of all doves are computed $j = (1, \dots N)$ in every epoch as the total number of crumbs in d-th dove's position. The fitness value is computed using equation (2) in this work.

In the next stage, selected doves are ranked with respect to fitness value, and $2N$ individuals are taken to perform the genetic operations. This enhances the performance of GDSO by increasing its searchability. Genetic operations such as crossover and mutation are performed to get the new search with the best fitness values. Here, $2N$ individuals are selected to perform mutation and crossover.

Here, the binomial crossover strategy combines the dove placed in the best and worst locations using the following steps. The i-th best dove is expressed as $H_i = (h_{i1}, h_{i2}, \dots ., h_{id})$ and its mutants are specified as $FM_i = (fm_{i1}, fm_{i2}, \dots ., fm_{id})$. These two terms are combined to get the crossover dove $FC_i = (fc_{i1}, fc_{i2}, \dots ., fc_{id})$. The performed crossover strategy is expressed as follows,

$$fc_{ij} = \begin{cases} fm_{ij} & if\ r_j \le CR\ or\ j = j_{max} \\ h_{ij} & if\ r_j > CR\ and\ j \ne j_{max} \end{cases} \quad (10)$$

Where r_j indicates the random number uniformly distributed in [0, 1], j_{max} represent the arbitrary integer lies between [1, d]. The crossover rate is denoted as CR which is utilized to determine the amount of genetic information that should be passed to crossover dove fc_{ij}. This crossover rate linearly decreases based on the iterations, and it is expressed as follows,

$$CR(k) = cr_{max} - (cr_{max} - cr_{min})\frac{k}{K} \quad (11)$$

Where, cr_{max} representing the maximum value of CR and cr_{min} representing the minimum value of CR, K indicates the number of iterations.

The dove's mutation strategy is expressed as,

$$fm(i,j) = M_{best_j} + Q.\left(h_{l_1,j} - h_{l_2,j}\right) + Q.\left(h_{l_3,j} - h_{l_4,j}\right) \quad (12)$$

Where, $fm(i,j)$ indicates the location of i-th mutant dove with respect to the aspect j, M_{best_j} indicates the location of the best dove obtained by performing a number of iterations. $l_1 \neq l_2 \neq l_3 \neq l_4 \neq i \in \{1,n\}$ denotes the randomly selected doves Q represents the function used to generate homogeneously distributed arbitrary numbers [0, 1]. The differences $\left(h_{l_1,j} - h_{l_2,j}\right)$ and $\left(h_{l_3,j} - h_{l_4,j}\right)$ are regulated utilizing the function Q as per the problem. This regulation ensures the optimal variability of the population in both local and global searches.

After executing the crossover and mutation operations, the newly formed doves are stored in the new solution set. This newly formed solution set is used for obtaining the optimal solution. Further, the fitness value is computed for the doves in the new solution set and ranked based on the fitness value. This process helps in finding the optimal solution.

After finding the set of new solutions, dove d_j^e is located near where the optimal solution is found are identified utilizing either maximum or minimum criterion at epoch e. This work focussed on minimizing the values of response time, expected cost, energy consumption and execution time. Therefore, a minimum criterion is taken for finding the optimal solution. It is represented as follows,

$$d_j^e = argmin\left\{f\left(w_j^e\right)\right\}, for\ j = 1,2 \ldots. N \quad (13)$$

Then, the satiety of each dove is updated using the below equation,

$$S_j^e = \begin{cases} \lambda S_j^{e-1} + e^{(f(w_j - f(w_{df}))}, & \text{if } f\left(w_{df}\right) \neq 0 \\ \lambda S_j^{e-1} + 1, & if\ \ f\left(w_{df}\right) = 0 \end{cases}, for\ j = 1,2 \ldots. N \quad (14)$$

The most satisfied dove d_j^e with the highest degree of satiety, the dove with the optimal solution is selected utilizing the minimum criterion. The selected dove in equation (15) has the best foraging behaviour, and other doves use this in the flocks to have the feed. This dove is taken to solve the multi-objective problem in TS. It is represented as,

$$d_s^e = \arg \max_{1 \le j \le N} \left\{S_j^e\right\}, for\ j = 1,2 \ldots N \quad (15)$$

Next, the position vector of each dove is updated utilizing the minimum criterion. In equation (16), the learning rate for updating the position vector of the dove is denoted as η. When the dove tries to follow the foraging behaviour of the best dove, it moves toward the position of the best dove to discover more food. This learning is simulated by modifying w_j^e, a position vector, and it could be the dove with the highest degree of satiety. The updation is expressed as,

$$\underline{w}_j^{e+1} = \underline{w}_j^e + \eta\beta_j^e\left(\underline{w}_{d_s}^e - \underline{w}_j^e\right) \quad (16)$$

Here, $\beta_j^e = \left(\frac{S_{b_s}^e - S_j^e}{S_{b_s}^e}\right)\left(1 - \frac{\left\|\underline{w}_j^e - \underline{w}_{d_s}^e\right\|}{maxDistance}\right) \quad (17)$

$$maxDistance: \max_{1 \le j \le N} \left\|\underline{w}_j^e - \underline{w}_{d_s}^e\right\| \quad (18)$$

Suppose the dove has a high degree of satiety. In that case, it does not change its foraging behaviour, whereas the dove with a low degree of satiety strongly desires to alter its foraging behaviour and is probably based on the best individual's behaviour. This change is done by adjusting $\left(\frac{S_{b_s}^e - S_j^e}{S_{b_s}^e}\right)$ of Eq. (17). Normally, doves' social impact degrades when it spreads out. Hence, the degree of impact is always inversely proportional to the distance between the best dove and other doves in the flock. This change is simulated by adjusting the amount proportional to the value of $\left(1 - \frac{\left\|\underline{w}_j^e - \underline{w}_{d_s}^e\right\|}{maxDistance}\right)$ in Eq. (17).

After the updation stage, the number of epochs is increased, which is given in Eq. (13), until it reaches the termination condition. The condition used for termination is expressed as follows,

$$\left|f_{d_s}^e - T(e)\right| \leq \varepsilon ore \leq set \max epoch \qquad (19)$$

Similarly, if the optimization is taken as the maximum criterion, which means the best solution is finding of maximum $\underline{w}_j^e$, then the equations (13) and (14) are changed as follows.

$$d_j^e = argmax\{f(w_j^e)\}, for\ j = 1,2 \ldots .N \qquad (20)$$

$$S_j^e = \lambda S_j^{e-1} + e^{(f(w_j - f(w_{df}))}, for\ j = 1,2 \ldots N \qquad (21)$$

By following the foraging behaviour of the dove, the multi-objective optimization problem in TS is resolved with the influence of operations of GA utilizing the proposed GDSO methodology. The multi-objective problems, such as $E_{ij}, TS_{Res}, ET,$ and TEC are solved for task scheduling, and the optimal solution is obtained, which enhances the performance of the cloud system.

Algorithm 1: GDSO for task scheduling

```
Determine VM's task numbers
Initialize population of Doves X
Set initial position, other parameters
Establish the fitness function utilizing Eq. (2)
Set Iterations $it = 0$,
For each Dove
Calculate the fitness values
Assign ranks to dove based on fitness value and arrange them in descending order
Select $2N$ individuals from the arrangement
Perform crossover, $fc_{ij} = \begin{cases} fm_{ij} & if\ r_j \leq CR\ or\ j = j_{max} \\ h_{ij} & if\ r_j > CR\ and\ j \neq j_{max} \end{cases}$
Perform mutation, $fm(i,j) = M_{best_j} + Q.\left(h_{l_1,j} - h_{l_2,j}\right) + Q.\left(h_{l_3,j} - h_{l_4,j}\right)$
Calculate the fitness value for the new solution using Eq. (2)
Locate the position of the dove with the optimal solution using Eq. (13)
Updation of satiety degree of dove using Eq.(14)
Selection of dove with high satiety degree using Eq.(15)
Update the position vector of other doves based on Eq.(16)
Increase the iterations to obtain the optimal solution
When the termination condition is satisfied, the process gets ended
Else it repeats the steps from assessing the fitness function
End
```

4. RESULTS AND DISCUSSION

The experiments are conducted to evaluate the performance of the proposed GDSO technique in task scheduling utilizing CloudSim 3.0.3 simulation tool. Multiple isolated tasks are created utilizing Planet lab workload traces to evaluate the performance of GDSO. The planet lab workload incorporates hundreds of VM’s CPU utilization data in which 500 servers worldwide are included. The experiments are conducted on Intel® core i5 processor with 1.8GHz CPU frequency, 8GB RAM and Windows 10 operating system using Netbeans and JDK 11. The configuration used in CloudSim for evaluating GDSO is presented in Table 1.

Table.1. Simulation parameter settings

Unit	Parameter	Value
Data Centre	Capacity	1
	Category	Heterogeneous
	Link delay (milliseconds)	10-100
	Bandwidth (Gbps)	2-20
Host	Number	10
	Cores	2-8
	RAM (MB)	2048
	Memory (MB)	500000
	bandwidth (bps)	5000
VM	Number	10-100
	CPU (MIPS)	2000
	RAM (MB)	1024
	Bandwidth (bps)	2000
	Cores per VM	1
Task	Count	2000
	Length of task (MI)	100-1000
	Size of task	100-500
	Iterations	10 - 30

The efficacy of the proposed model is measured in terms of Total expected cost (TEC), Execution time, Response time and energy consumption. The values obtained by these measures for 20 VM’s data with multiple numbers of tasks are presented in table 2.

Table.2. Performance Outcome of proposed GDSO (VMs= 20)

Number of tasks	Energy Consumption (J)	Total Expected Cost ($)	Execution Time (ms)	Response time (seconds)	CPU Utilization (%)	Bandwidth Utilization (%)
25	0.4318	37	3.9	0.002421	29.564	6.91
50	0.6801	64	6.98	0.004263	43.478	9.01
75	1.3734	84	8.9	0.004721	55.685	10.32
100	2.4751	111	11.3	0.00618	70.498	11.96

The outcome of GDSO in 20 VM data shows that the consumption of energy increases with the number of tasks executed. Likewise, the Execution time, total expected cost and response time of the proposed GDSO technique is increased when the number of tasks is increased. The energy consumption of GDSO in 25 tasks is 0.4318 J, increasing to 57.50% for 50 tasks, 218.06% for 75 tasks and 473.20% for 100 tasks. The TEC for 25 tasks is 37$, which increases to 72.97$ for 50 tasks, 127.02% for 75 and 200% for 100 tasks. The CPU utilization for 25 tasks is 29.564%, increasing to 13.91% for 50 tasks, 26.12% for 75 and 41% for 100 tasks. The bandwidth utilization of 25 tasks is 6.91% which increases upto 2.1%, 3.41% and 5.05% for 50, 75 and 100 tasks. ET for 25 tasks is 3.9 s which is 3.08ms, 5ms, and 7.4ms less compared to 50, 75 and 100 tasks. The response time taken for 25 tasks is 0.002421, which is 0.001842s, 0.0023s and 0.03759 less compared to 50, 75 and 100 tasks.

Table.3. Performance Outcome of proposed GDSO (VMs= 100)

Number of tasks	Energy Consumption (J)	Total Expected Cost ($)	Execution Time (ms)	Response time (seconds)	CPU Utilization (%)	Bandwidth Utilization (%)
200	6.93	197	22.32	0.0132	78.496	16.87
400	11.753	232	23.99	0.01553	81.124	21.387
600	12.301	255	27.89	0.0185	83.458	25.489
800	17.457	275	35.0	0.02275	87.49	29.897
1000	24.853	301	42.16	0.02734	90.01	32.036

Table 3 represents the outcome of GDSO while evaluating 100 VMs data. Here, the number of tasks, such as 200, 400, 600, 800, and 1000 are evaluated. The energy consumed by the proposed GDSO in 200 tasks with 100 VMs is 6.9J which increases upto 69.59%, 151.90%, and 258.62% for 400, 600,800 and 1000 tasks. Likewise, TEC for 200 tasks is 197$, increasing by 17.76%, 29.44%, 39.59% and 52.79%. CPU utilization for 200 tasks is 78.496%, increasing by 2.62%, 4.962%, 8.99% and 11.51%. Bandwidth utilization for 200 tasks is 16.87%, increasing to 4.51%, 8.61%, 13.02% and 15.16%. The execution time for 200 tasks is 22.32ms which is 1.67ms, 5.57ms, 12.68ms and 19.84ms less. The response time taken for 200 tasks is 0.0132, which is 0.00233s, 0.0053s, 0.00955s and 0.01414s less than 400, 600,800 and 1000 tasks. The performance measures gained effective results while executing with less number of tasks. But, the values of performance measures increase proportionally to the number of tasks. However, the rise of values in E_{ij}, TS_{Res}, ET, and TEC are considerably low compared to the existing approach. This shows the superiority of the proposed GDSO model.

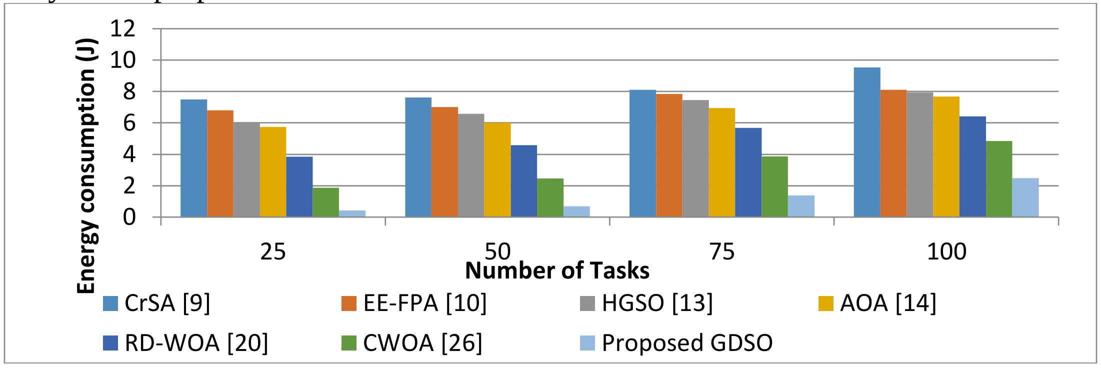

Figure.2. Comparison of Energy Consumption

Figure 2 displays the energy consumption comparison of the task scheduling process in the proposed GDSO and existing methodologies. The energy consumption analysis of 20 VMs is given. The energy consumed by the proposed GSDSO is 2.4751J which is about 74.05%, 69.45%, 68.86%, 67.77%, 61.38%, and 49.03% reduction compared to prevailing TS models such as CrSA, EE-FPA, HGSO, AOA, RD-WOA, and CWOA for executing 100 tasks. The result indicates that energy consumed by GDSO in the task scheduling process is too low compared to existing approaches.

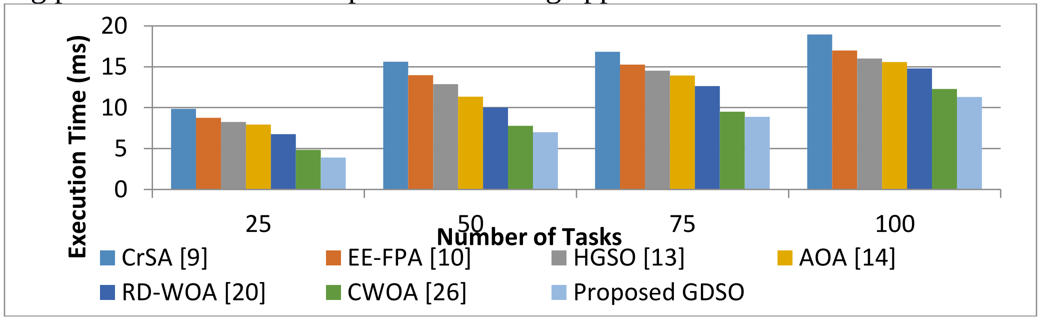

Figure.3. Comparison of Execution Time

Figure 3 illustrates an execution time comparison between the proposed GDSO and existing approaches. The proposed GDSO executes 100 tasks in 11.3ms, 7.65ms, 5.68ms, 4.72ms, 4.29ms, 3.5ms, and 1ms lower than CrSA, EE-FPA, HGSO, AOA, RD-WOA, and CWOA methods. The execution time is low in executing 25 tasks but increases with respect to the count of the tasks.

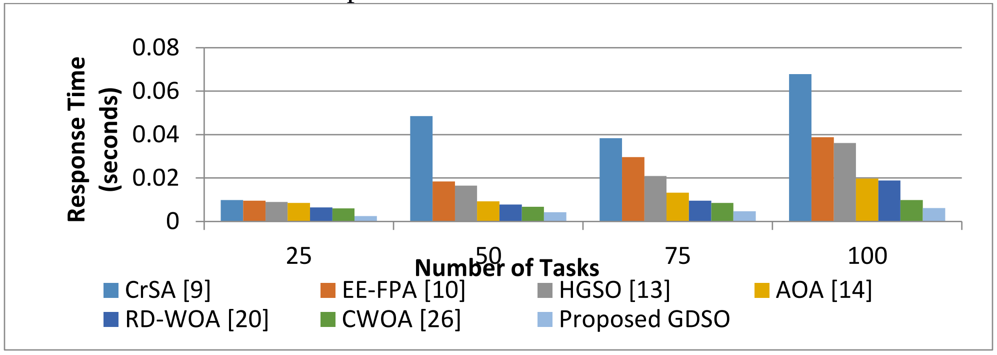

Figure.4. Comparison of Response Time

Figure 4 represents the response time comparison of the proposed GDSO and existing methods. The response time of 20 VM data is taken for comparison. The response time taken for processing 100 tasks by proposed GDSO is 0.00618s, and it is 0.06162s, 0.03262s, 0.03002s, 0.01352s, 0.01272s and 0.00369s less than CrSA, EE-FPA, HGSO, AOA, RD-WOA, and CWOA models. Like the other performance measures, the response time increases based on task counts. The response time of executing 25 tasks is low compared to executing 50, 75 and 100 tasks, but it gradually increases based on tasks.

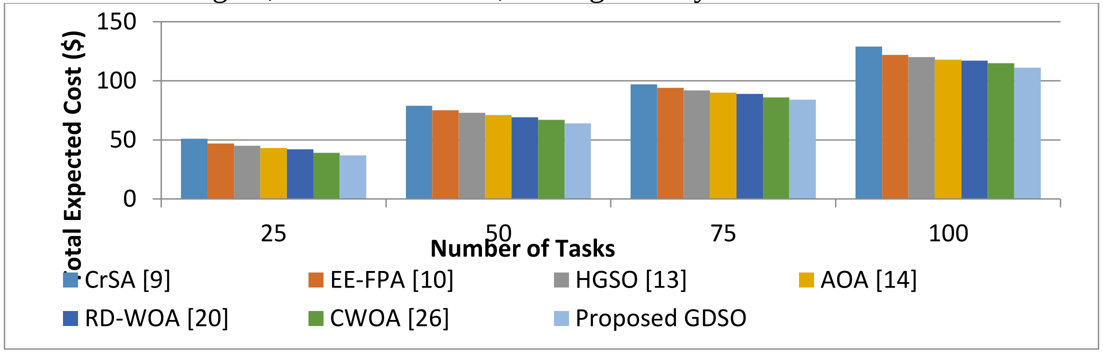

Figure.5. Comparison of Total expected cost (TEC)

Figure 5 displays the TEC analysis of suggested GDSO and existing approaches. The TEC of the proposed GDSO model is less compared to other approaches. The figure shows the cost analysis of 20 VMs with 25, 50, 75 and 100 tasks. While observing the outcome, the cost of the task scheduling process increases with the number of tasks. The TEC of the proposed GDSO is 111$, about 13.95%, 9.01%, 7.5%, 5.9%, 5.12%, and 3.47% reduction than CrSA, EE-FPA, HGSO, AOA, RD-WOA, and CWOA approaches.

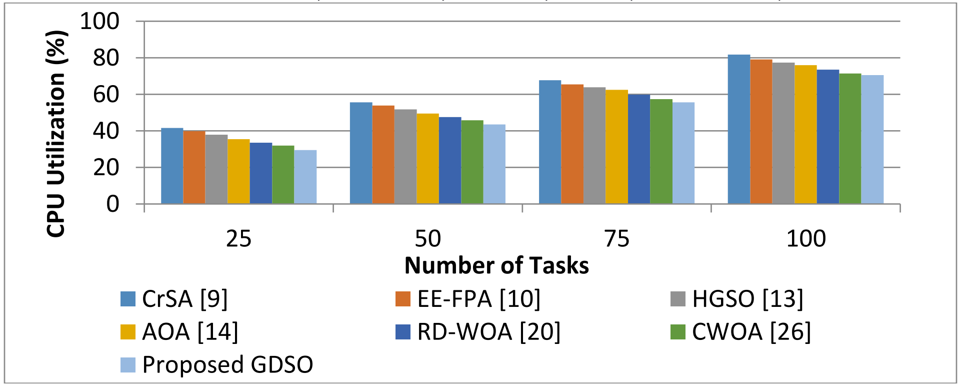

Figure.6. CPU Utilization

Figure 6 compares CPU utilization of proposed GDSO and prevailing task scheduling approaches. The CPU utilization of the proposed GSDO model is less compared to existing methods. The proposed GDSO attained the CPU utility of 70.49% in 100 tasks with 20 VMs. The CPU utilization decreased by 11.3% compared to CrSA, 8.66% for EE-FPA, 6.83% for HGSO, 5.49% for AOA, 2.96% for RD-WOA, and 0.96% for CWOA.

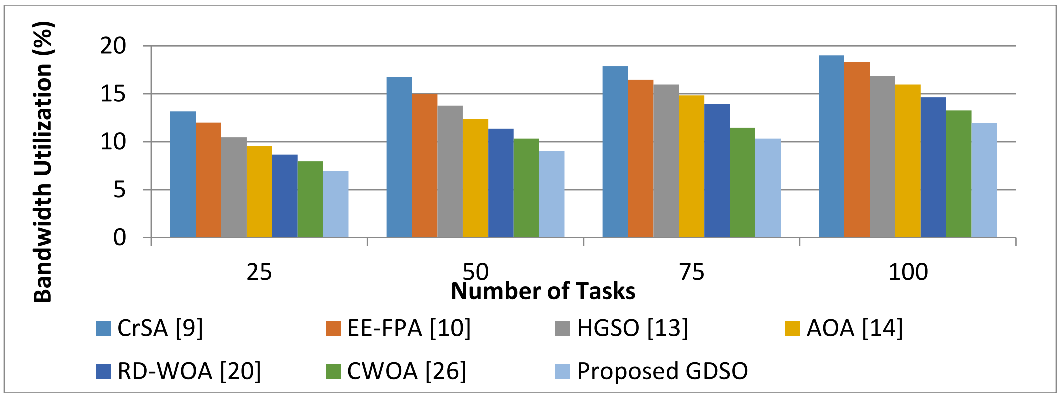

Figure.7. Bandwidth Utilization

Figure 7 presents the Bandwidth utilization of suggested GDSO and existing task scheduling methods. The GDSO method uses reduced bandwidth compared to other methods. The bandwidth utilization of the proposed GDSO is 11.96% while processing 100 tasks in 20 VMs. It is about 7.05%, 6.36%%, 4.89%, 4%, 2.69%, and 1.3% reduction compared to the existing methods such as CrSA, EE-FPA, HGSO, AOA, RD-WOA, and CWOA.

The performance evaluation of the proposed GDSO indicates that it achieved better performance in TS compared to existing approaches. This method solves the execution delay problem and enhances the TS process's efficiency. Hence, the GDSO technique could be utilized for an effective task-scheduling process.

5. CONCLUSION

This paper presented an effective task-scheduling approach to solve the execution delay problem. The proposed task scheduling approach uses Genetic Dove Swarm Optimization (GDSO), which is constructed by combining GA and DSO approaches. The GDSO solved the multi-objective problem and enhanced the efficacy of the task-scheduling process. This model is simulated utilizing the CloudSim tool. The evaluation result shows that the suggested GDSO model has achieved better performance than existing approaches. The performance indicators such as energy consumption, execution time, response time, total expected cost, CPU utilization and bandwidth utilization are used for evaluation. In future, the possibility of enhancing the security and reliability in scheduling trusted tasks will be investigated. Also, the possibility of reducing task execution time through parallel processing will be examined.

Reference

1. Abdullahi, M., & Ngadi, M. A. (2016). Symbiotic organism search optimization based task scheduling in cloud computing environment. *Future Generation Computer Systems*, *56*, 640-650.
2. Natesan, G., & Chokkalingam, A. (2019). Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm. *ICT Express*, *5*(2), 110-114.
3. Awad, A. I., El-Hefnawy, N. A., & Abdel_kader, H. M. (2015). Enhanced particle swarm optimization for task scheduling in cloud computing environments. *Procedia Computer Science*, *65*, 920-929.
4. Sreenu, K., & Sreelatha, M. (2019). W-Scheduler: whale optimization for task scheduling in cloud computing. *Cluster Computing*, *22*(1), 1087-1098.

5. Wei, X. (2020). Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, 1-12.
6. Zhan, S., & Huo, H. (2012). Improved PSO-based task scheduling algorithm in cloud computing. *Journal of Information & Computational Science*, *9*(13), 3821-3829.
7. Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., & Murphy, J. (2020). A WOA-based optimization approach for task scheduling in cloud computing systems. *IEEE Systems Journal*, *14*(3), 3117-3128.
8. Gawali, M. B., & Shinde, S. K. (2018). Task scheduling and resource allocation in cloud computing using a heuristic approach. *Journal of Cloud Computing*, *7*(1), 1-16.
9. Prasanna Kumar, K. R., & Kousalya, K. (2020). Amelioration of task scheduling in cloud computing using crow search algorithm. *Neural Computing and Applications*, *32*(10), 5901-5907.
10. Bezdan, T., Zivkovic, M., Antonijevic, M., Zivkovic, T., & Bacanin, N. (2021). Enhanced flower pollination algorithm for task scheduling in cloud computing environment. In *Machine learning for predictive analysis* (pp. 163-171). Springer, Singapore.
11. Rjoub, G., Bentahar, J., Abdel Wahab, O., & Saleh Bataineh, A. (2021). Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurrency and Computation: Practice and Experience*, *33*(23), e5919.
12. Velliangiri, S., Karthikeyan, P., Xavier, V. A., & Baswaraj, D. (2021). Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Engineering Journal*, *12*(1), 631-639.
13. Abd Elaziz, M., & Attiya, I. (2021). An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing. *Artificial Intelligence Review*, *54*(5), 3599-3637.
14. Abualigah, L., & Diabat, A. (2021). A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Cluster Computing*, *24*(1), 205-223.
15. Gupta, S., Iyer, S., Agarwal, G., Manoharan, P., Algarni, A. D., Aldehim, G., & Raahemifar, K. (2022). Efficient prioritization and processor selection schemes for heft algorithm: A makespan optimizer for task scheduling in cloud environment. *Electronics*, *11*(16), 2557.
16. Bal, P. K., Mohapatra, S. K., Das, T. K., Srinivasan, K., & Hu, Y. C. (2022). A Joint Resource Allocation, Security with Efficient Task Scheduling in Cloud Computing Using Hybrid Machine Learning Techniques. *Sensors*, *22*(3), 1242.
17. Abdullahi, M., Ngadi, M. A., Dishing, S. I., & Abdulhamid, S. I. M. (2022). An adaptive symbiotic organisms search for constrained task scheduling in cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, 1-12.
18. Amer, D. A., Attiya, G., Zeidan, I., & Nasr, A. A. (2022). Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. *The Journal of Supercomputing*, *78*(2), 2793-2818.
19. Bezdan, T., Zivkovic, M., Bacanin, N., Strumberger, I., Tuba, E., & Tuba, M. (2022). Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm. *Journal of Intelligent & Fuzzy Systems*, *42*(1), 411-423.
20. Manikandan, N., Gobalakrishnan, N., & Pradeep, K. (2022). Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment. *Computer Communications*, *187*, 35-44.
21. Rajakumari, K., Kumar, M. V., Verma, G., Balu, S., Sharma, D. K., & Sengan, S. (2022). Fuzzy Based Ant Colony Optimization Scheduling in Cloud Computing. *Comput. Syst. Sci. Eng.*, *40*(2), 581-592.
22. Abualigah, L., & Alkhrabsheh, M. (2022). Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. *The Journal of Supercomputing*, *78*(1), 740-765.

23. Imene, L., Sihem, S., Okba, K., & Mohamed, B. (2022). A third generation genetic algorithm NSGAIII for task scheduling in cloud computing. *Journal of King Saud University-Computer and Information Sciences*.
24. Nabi, S., Ahmad, M., Ibrahim, M., & Hamam, H. (2022). AdPSO: adaptive PSO-based task scheduling approach for cloud computing. *Sensors*, *22*(3), 920.
25. Mahmoud, H., Thabet, M., Khafagy, M. H., & Omara, F. A. (2022). Multi-objective Task Scheduling in Cloud Environment Using Decision Tree Algorithm. *IEEE Access*, *10*, 36140-36151.
26. Pradeep, K., Ali, L. J., Gobalakrishnan, N., Raman, C. J., & Manikandan, N. (2022). CWOA: hybrid approach for task scheduling in cloud environment. *The Computer Journal*, *65*(7), 1860-1873.
27. Kruekaew, B., & Kimpan, W. (2022). Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning. *IEEE Access*, *10*, 17803-17818.