# Deadline-aware Task Scheduling for Cloud Computing using Firefly Optimization Algorithm

BAI Ya-meng*, WANG Yang, WU Shen-shen
School of Information Engineering, Jiaozuo University, Jiaozuo 412000, China

***Abstract*—Task scheduling poses a major challenge for cloud computing environments. Task scheduling ensures cost-effective task execution and improved resource utilization. It is classified as a NP-hard problem due to its nondeterministic polynomial time nature. This characteristic motivates researchers to employ meta-heuristic algorithms. The number of cloud users and computing capabilities is leading to increased concerns about energy consumption in cloud data centers. In order to leverage cloud resources in the most energy-efficient manner while delivering real-time services to users, a viable cloud task scheduling solution is necessary. This study proposes a new deadline-aware task scheduling algorithm for cloud environments based on the Firefly Optimization Algorithm (FOA). The suggested scheduling algorithm achieves a higher level of efficiency in multiple parameters, including execution time, waiting time, resource utilization, the percentage of missed tasks, power consumption, and makespan. According to simulation results, the proposed algorithm is more effective and superior to the CSO algorithm under HP2CN and NASA workload archives.**

***Keywords—Cloud computing; energy efficiency; task scheduling; firefly algorithm***

## I. Introduction

In recent years, wireless and emerging technologies have undergone significant progress, particularly the Internet of Things (IoT) [1, 2], artificial intelligence [3], machine learning [4-6], smart grids [7], Blockchain [8], 5G connectivity [9], and cloud computing [10], resulting in a number of positive effects on society. Cloud computing offers convenient, flexible, and ubiquitous access to a set of configurable computing resources, such as servers, storage, applications, and services, which are delivered via the web and released instantly [11, 12]. It allows users to access computer resources without having to manage them actively [13]. In this regard, three types of services can be provided by a cloud: infrastructure, software, and platform [14]. The first service is infrastructure as a service (IaaS), which offers storage and computational resources [15]. In the second service, users can access the software remotely without installing it locally, which is called software as a service (SaaS) [16, 17]. The third service is the platform as a service (PaaS), which provides a platform on which clients can build applications [18]. Virtual Machines (VMs) are provided by cloud providers as computing resources. To improve the efficiency of cloud computing, optimal task scheduling is critical when numerous users demand services from the cloud [19].

To achieve the desired quality of service (QoS), efficient task scheduling allocates resources optimally across the desired tasks in a timely manner. Optimizing a given objective involves building a schedule of tasks to be allocated to VMs with consideration of some constraints [20]. Each cloud infrastructure relies on a task scheduling algorithm as a key component. The performance metrics used in scheduling procedures involve computation-based indicators, including response time, energy consumption, and makespan, and network-based indicators, including round trips, communication cost, and traffic volume [21]. There are three types of optimal task scheduling approaches in cloud computing: heuristic, meta-heuristic, and hybrid. Heuristic task scheduling algorithms offer ease of scheduling and deliver the best possible solution, but they do not guarantee optimal outcomes. A meta-heuristic approach can find optimal solutions to task scheduling problems in a polynomial amount of time. The hybrid task scheduling algorithm combines both the heuristic and meta-heuristic approaches [22, 23].

Although there are promising approaches to efficient task scheduling in the cloud, the problem of task scheduling remains NP-complete [24, 25]. This paper proposes and uses the Firefly Optimization Algorithm (FOA) in order to schedule a bag of tasks in a cloud environment, avoiding network communication and data transfer costs. In the experimental setup, the proposed algorithm is compared to the Highest Response Ratio Next (HRRN), Shortest Process Next (SPN), First Come First Served (FCFS), and PSO algorithms. The proposed method was tested using various distributions to gain insight into its performance trend. By optimizing both the time overhead and the energy consumption of the mobile device, the QoS for mobile users is enhanced by maximizing the overall system benefits. The main contributions of this paper are as follows:

- Defining the task scheduling problem and formulating mathematical models and the objective functions used to optimize the allocation of tasks to virtual machines.
- Analyzing the performance of the proposed algorithm in terms of execution time, makespan, and energy consumption.
- Verifying the experimental results by comparing them to CSO, FCFS, and PSO findings.

The remainder of the paper follows in the following order. A discussion of recent cloud task scheduling methods is presented in Section II. Section III describes and models the problem of task scheduling in a cloud computing environment, followed by an explanation of the proposed algorithm. Section IV discusses the simulation results. The paper concludes with Section V.

## II. Related Work

Yu and Su [26] proposed a system called Three Queues (TQ) that uses dynamic priorities and three queues in order to cope with the increasing heterogeneity of cloud computing clusters. Based on the priority of tasks, the algorithm places tasks in a waiting queue and then categorizes them based on the input amount, output amount, number of current tasks running on a node, completion time, and disk I/O rate of the Map phase. Hardware utilization is improved by placing jobs in corresponding queues. It has been demonstrated that this algorithm improves task scheduling performance in the presence of both CPU-intensive and I/O-intensive tasks and shortens task execution times. The genetic algorithm proposed by Sun, et al. [27] uses phagocytosis as a crossover operation, generates a sub-chromosomal individual by phagocytosing two mother chromosomes, produces a random third individual, and determines a new individual resulting from phagocytosis based on the load-balancing standard deviation and fitness factor, which results in a high percentage of high-quality individuals. An evolutionary genetic algorithm for multiple populations is then used that creates initial subpopulations using the Min-Min algorithm, and these subpopulations are evolved using an improved genetic algorithm. According to simulations, the proposed algorithm schedules cloud tasks efficiently.

Al-Maytami, et al. [28] developed a new scheduling algorithm using Directed Acyclic Graphs (DAG) and Prediction of Tasks Computation Time (PTCT). Furthermore, by reducing the Expected Time to Compute (ETC) matrix using Principle Components Analysis (PCA), the proposed algorithm significantly improves makespan and minimizes complexity and computation. According to simulation results, the algorithm outperforms other algorithms for heterogeneous systems regarding schedule length rate, response time, and efficiency. Prasanna Kumar and Kousalya [29] used the Crow Search Algorithm (CSA) to schedule cloud tasks. The CSA draws inspiration from the food-collecting behavior of crows. The crow keeps on searching for a better food source than its current food source as it keeps watching its mates. This paper uses the CSA to find suitable VMs and minimize the makespan. CloudSim is used to measure CSA's performance over ACO algorithms. Simulation results indicate that the CSA algorithm is superior to the ACO algorithms.

Panda and Jana [16] developed an energy-efficient task scheduling algorithm (ETSA) to resolve task scheduling problems. The algorithm considers and normalizes both the completion time and the total resource utilization of a task. ETSA was evaluated for its ability to measure energy efficiency and makespan in heterogeneous environments. ETSA was tested in a wide variety of heterogeneous environments, and the test results showed that it was able to achieve better energy efficiency and makespan than existing algorithms. The algorithm also accounts for both completion time and resource utilization, which gives it an advantage over other scheduling algorithms. Na, et al. [30] propose a Squid operator and Nonlinear Inertia Weight PSO (SNW-PSO) algorithm to better meet the users' QoS requirements in cloud computing. Execution cost and execution time are optimized by the algorithm. As part of its optimization process, nonlinear inertia weights are introduced to prevent the algorithm from jumping from its local optimum. Furthermore, the squid operator allows for greater particle diversity and faster convergence to optimal positions within particle swarms. This algorithm compensates for the weaknesses of the traditional PSO algorithm, namely its ease of over-convergence and tendency towards the local optimum. The SNW-PSO algorithm converges more quickly than an LDIC-PSO algorithm and reduces both task completion time and cost when compared to a PSO algorithm with linearly decreasing inertia weight.

## III. Proposed Method

In cloud computing, different quality of service parameters is optimized by scheduling tasks. The task scheduling problem entails allocating several tasks appropriate to a certain number of VMs. This section proposes an evolutionary task scheduling approach using FOA. Environmental dynamics, deadlines, and declining the entire task are the basis of the proposed method. The task scheduling problem is modeled based on the following assumptions:

- The VMs are heterogeneous in terms of processing power and power consumption;
- There is no migration of tasks between VMs;
- All submitted tasks are independent.

### A. FOA Formulation for Task Scheduling Issue

The firefly algorithm is a novel technique based on fireflies' social behaviors in nature. They flash short lights rhythmically. Each one's flashing pattern is unique and different from the rest. They utilize the lights for the mate-attracting process and attract prey. Furthermore, these lights can function as a protective mechanism in favor of fireflies. The rhythmic light, flashing rate, and time interval between flashing signals cause the two sexes to get attracted to each other. Each parcel is a firefly which is updated according to a firefly's awareness of its neighbors in the multidimensional search space through attracting dynamically. The parameters of FOA are described in the following.

- Light intensity factor (I): In the firefly algorithm, the light intensity factor is defined by Eq. 1. In this equation, $I_0$ denotes the initial light, $r$ is the distance between two fireflies, and $I$ is the received light intensity.

$$\mathrm{I} = \frac{\mathrm{I_0}}{\mathrm{r^2}} \tag{1}$$

- Attractiveness: The amount of a firefly's attractiveness corresponds to the light intensity that neighboring firefly witnesses and is defined by Eq. 2 and Eq. 3. The parameter $\beta$ is used for measuring the attractiveness (attraction) between two fireflies.

$$\beta = \beta_0 \times e^{-\gamma r} \tag{2}$$

$$\beta = \frac{\beta_0}{1 + \gamma r^2} \tag{3}$$

- The distance between fireflies: Euclidean distance determines the distance between two fireflies, calculated as follows.

$$r_{ij} = \|x_i - x_j\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{4}$$

- Luciferin release: Like other algorithms, the firefly algorithm starts with a randomly selected population. Therefore, the optimization begins randomly with a population of *n* fireflies in the search space. According to Eq. 5, the algorithm first updates the luciferin share of every firefly and then updates their position in every repetition.

$$l_j(t+1) = (1-\rho)l_j(t) + \gamma j_j(t+1) \tag{5}$$

In each repetition, the luciferin amount for every firefly is determined according to the fitness of its position. It means, based on the amount of fitness, some luciferin is added to the previously available amount for every repetition. In the above equation, $j_j(t+1)$ represents the fitness function (the fitness of the position) of the *i-th* firefly in *t* repetition of the algorithm, $\gamma$ is the luciferin rise constant, and $(1-\rho)l_j(t)$ is the amount of luciferin decrease.

- The probability of firefly selection: for every firefly *i*, the probability of turning toward the brighter neighbor *j* is expressed as follows. In Eq. 6, *t* denotes the time measure, $d_{i.j}(t)$ is the distance between two fireflies, $r_d^i(t)$ is the decision-making (intuitive) radius of a firefly, and *N(t)* is the ensemble of neighboring fireflies of the firefly *i* at time *t*.

$$P_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N(t)} l_k(t) - l_i(t)} \quad j \in N(t). N(t) = \{j: d_{i.j}(t) < r_d^i(t).\ l_i(t) < l_j(t)\} \tag{6}$$

- Updating the firefly position (a novel solution): The time-district moving of a firefly could be presented as follows:

$$x_i(t+1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}\right) \tag{7}$$

In the above equation, $x_i(t)$ refers to the m-dimension vector of the firefly *i* at time *t*, and s stands for the number of moving steps.

### *B. Task Scheduling Algorithm based on Discrete FOA*

The original purpose of the firefly algorithm was to solve continuous optimization problems, so it may not be effectively employed for discrete optimization problems. Therefore, in this paper, the Smallest Position Value (SPV) rule proposed by Bean [31] is implemented for the discrete firefly algorithm. The implementation is as follows:

*1) Solution representation:* The search space has *n* dimensions corresponding to the tasks' number; thus, every dimension represents one task. The vector $x_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_n^t)$ denotes the fireflies' positions in the search space. The SPV rule assigns tasks based on their positions. Fig. 1 illustrates an acyclic graph structure relevant to this research, consisting of six tasks. Table I presents the scheduling solutions. As dimension 4 in Table I has the smallest position value, it is the first task to be assigned, and dimension three is the second task, and so on. Initial population: Discrete firefly algorithms generate the initial population using a uniform distribution, similar to most meta-heuristic algorithms that generate the initial population randomly. The position values are also generated randomly by applying uniform random numbers to intervals of [0, 1].

*2) Solutions update:* Every firefly is evaluated using this permutation to determine its fitness function value. Every firefly's fitness function value is influenced by light intensity. Dim fireflies are attracted to bright ones. The firefly's attractiveness is determined by Eq. 2 and Eq. 3. The distance between two fireflies is measured by Eq. 4, and tasks are assigned using the SPV rule. Every firefly's attractiveness is calculated and then based on this value, its movement is determined by Eq. 7. The mentioned steps are repeated until the completion term is met when all the fireflies are attracted. To calculate the number of missed tasks, the beginning time and deadline of tasks are taken into account. In task scheduling, the main goal is to minimize the total execution time and the number of missed tasks. Eq. 8 calculates the fitness value of the proposed method for reducing total task execution time and the number of missed tasks. The input parameters are also normalized using Eq. 9.

Fitness=w_1×Makespan+w_2×Missed Task

w_1+w_2=1 (8)

Normalized(m)=(m-M_min)/(M_max-M_min ) (9)

Eq. 9 normalizes the parameter m between the largest and smallest values. Both cases were used in simulations to obtain the total execution time and the number of missed tasks. As shown in Eq. 10, the proposed method uses the tasks' completion times to determine the tasks' total execution times.

Makespan=min(C_(max$f_0$)).whereC_max>C_t .t=1.2.3.. n (10)

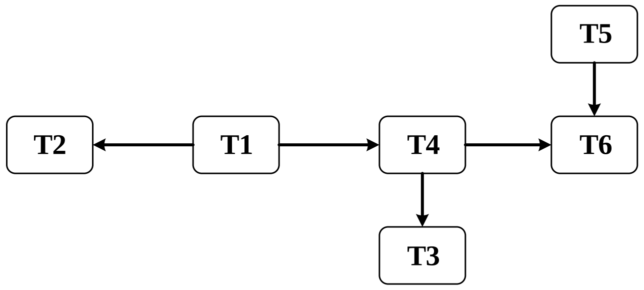

Fig. 1. DAG graph for six tasks.

TABLE I. THE SOLUTION EXAMPLE FOR SIX TASKS

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $x_{ik}^{t}$ | 0.3 | 0.4 | 0.092 | 0.035 | 0.59 | 0.61 |
| **Tasks** | T3 | T4 | T2 | T1 | T5 | T6 |

In the equation, t denotes a task, cmax is the maximum time required to complete it, and ct is the completion time of the task t.

## C. Solving an Example

In this subsection, the proposed method procedure is illustrated using an example of 6 VMs and 30 tasks whose durations are 100 and 200, and their deadlines are determined according to Table II. The completion time is 1.53 seconds, and the number of missed tasks' is 1. It should be noted that the initial attraction intensity for every firefly is randomly determined based on the Gaussian distribution and initial position of fireflies. Every firefly is assessed to calculate the fitness function value. The fitness function value for every firefly is associated with light intensity. Dim fireflies are attracted to bright ones. The firefly's attractiveness is determined by Eq. 3, and the distance between two fireflies is measured by Eq. 4. Tasks are assigned using the SPV law. Every firefly's attractiveness is calculated, and then based on this value, its movement is determined by Eq. 7. The mentioned steps are repeated until the completion term is met when all the fireflies are attracted. The proposed method's way of assigning and scheduling tasks is presented in Fig. 2.

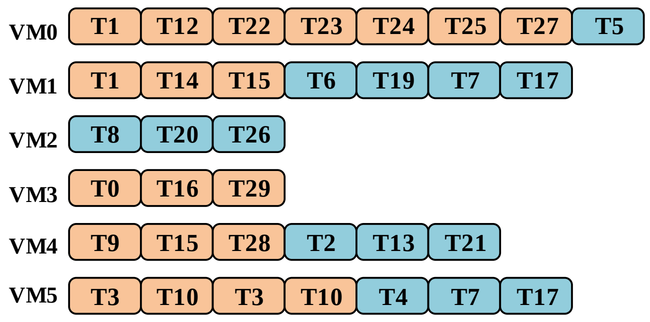

Fig. 2. Gantt chart for 30 tasks and six VMs.

TABLE II. DETAILS OF 30 TASKS

| #Task | Arrival Time | service time | Deadline | Dependency | #Task | Arrival Time | service time | Deadline | Dependency | #Task | Arrival Time | service time | Deadline | Dependency |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 14 | 4 | 8 | - | 11 | 15 | 9 | 18 | 5,3 | 21 | 17 | 9 | 18 | 8 |
| 2 | 15 | 2 | 4 | - | 12 | 9 | 9 | 18 | 10 | 22 | 11 | 6 | 12 | 21 |
| 3 | 19 | 5 | 10 | 1 | 13 | 3 | 7 | 14 | - | 23 | 20 | 6 | 12 | 19,29 |
| 4 | 14 | 8 | 16 | 1 | 14 | 4 | 3 | 6 | 11 | 24 | 6 | 6 | 12 | 16,19,27 |
| 5 | 9 | 9 | 18 | - | 15 | 5 | 5 | 10 | 16 | 25 | 11 | 4 | 8 | 16 |
| 6 | 9 | 2 | 4 | 2 | 16 | 7 | 4 | 8 | 15,2 | 26 | 10 | 7 | 14 | 23 |
| 7 | 6 | 1 | 2 | 5 | 17 | 8 | 3 | 6 | 15,11 | 27 | 9 | 9 | 18 | 17 |
| 8 | 18 | 8 | 16 | - | 18 | 2 | 1 | 2 | - | 28 | 18 | 4 | 8 | 16 |
| 9 | 20 | 3 | 6 | 1,7 | 19 | 13 | 7 | 14 | 8,9,13 | 29 | 4 | 1 | 2 | - |
| 10 | 2 | 5 | 10 | 2,5,7 | 20 | 5 | 8 | 16 | 5 | 30 | 13 | 4 | 8 | 31 |

## IV. EXPERIMENTAL RESULTS

Cloudsim is used to implement the proposed algorithm for task scheduling. The proposed method was simulated and compared to the available algorithms using the data in Table II [32]. In the proposed method, the entrance times of all tasks are equal, but the serving times, deadlines, and the tasks' interdependencies are considered according to Table II. For example, task 9 requires three units of time to get served, and its deadline to complete execution is 6. Also, tasks 1 and 7 should be completed prior to task 9. Table III presents the tasks' characteristics, Table IV contains the features of VMs, Table V outlines the features of data centers, and Table VI shows the parameters of the firefly algorithm.

### A. First Experiment: Evaluation of the Proposed Method Compared to [32]

The experiment simulates eight cloud resources and creates and executes thirty tasks, as shown in Table II. The tasks are designed to emulate real-world workloads and evaluate the performance of cloud resources. The experiment results are then used to analyze the performance of various cloud resource configurations. In order to examine the efficiency of the proposed algorithm, we compared it with HRRN, SPN, FCFS, and PSO algorithms. Fig. 3 to 6 illustrate that our method outperforms others in terms of overall execution time, average service time + waiting time, percentage of missed tasks, and resource utilization. The results show that our proposed algorithm can effectively optimize resource utilization and reduce the overall execution time of the cloud system. This is beneficial for cloud computing users, as it can reduce their costs and improve their performance. Furthermore, our proposed algorithm also provides more efficient scheduling and task assignment, resulting in better task scheduling, faster task completion, and higher resource utilization. This improved performance leads to better user experience and satisfaction.

TABLE III. TASKS' CHARACTERISTICS

| Parameters | Values |
|---|---|
| Task length | 1 – 100 |
| Input size | 300 |
| Output size | 300 |
| Number of processors | 1 |

TABLE IV. FEATURES OF VMS

| Parameters | Values |
|---|---|
| MIPS | 80-500-750-100 |
| Number of processors | 1 |
| RAM | 128 |
| Bandwidth | 2500 |

TABLE V. DATA CENTERS' FEATURE

| Parameters | Values |
|---|---|
| Speed | 500000 |
| Physical machine capacity | 10000 |
| Storage capacity | 1000000 |
| MIPS Bandwidth | 100000 |

TABLE VI. FIREFLY ALGORITHM PARAMETERS

| Parameters | Values |
|---|---|
| β | Min: 0, Max: 1, Mean: 0.5 |
| γ | Min: 0.5, Max: 1, Mean: 0.75 |
| A | Min: 0, Max: 1, Mean: 0.5 |

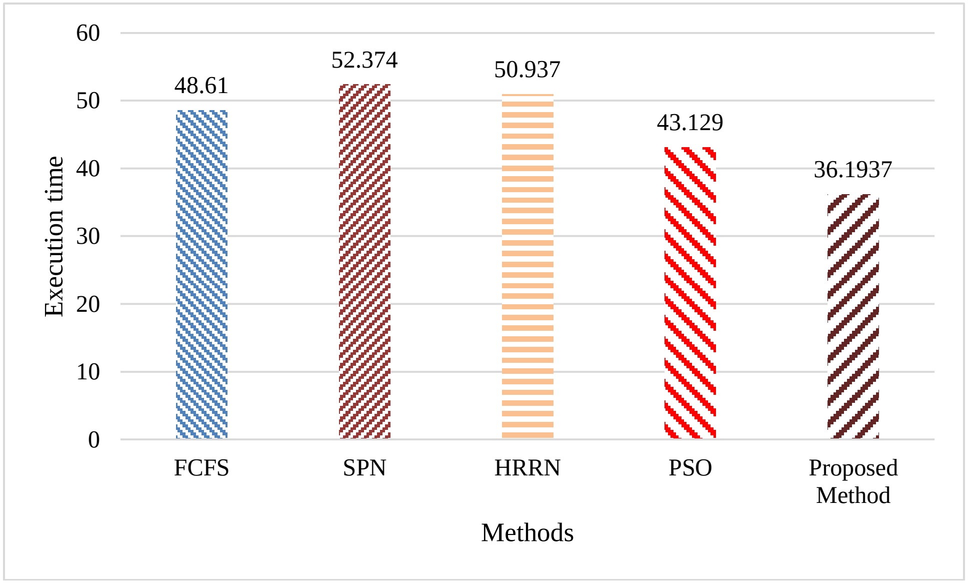

Fig. 3. Execution time comparison.

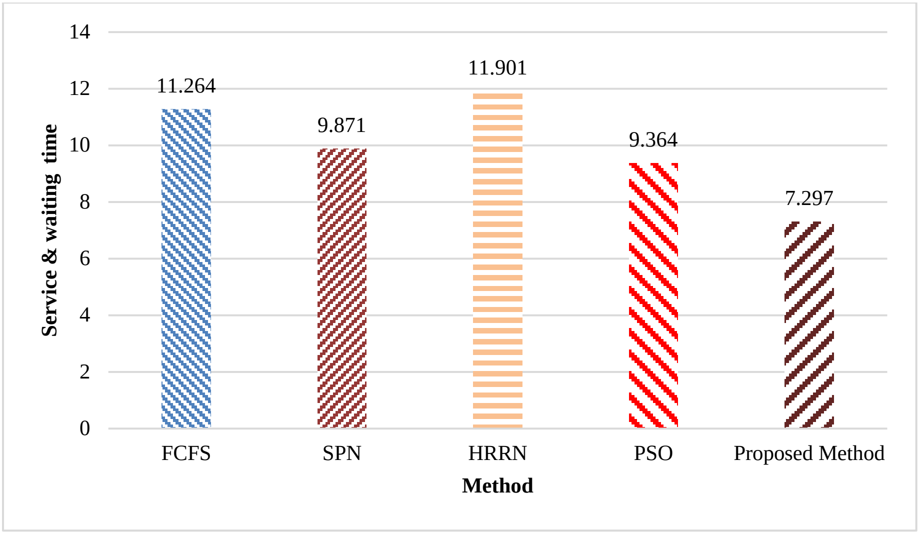

Fig. 4. Service and waiting time comparison.

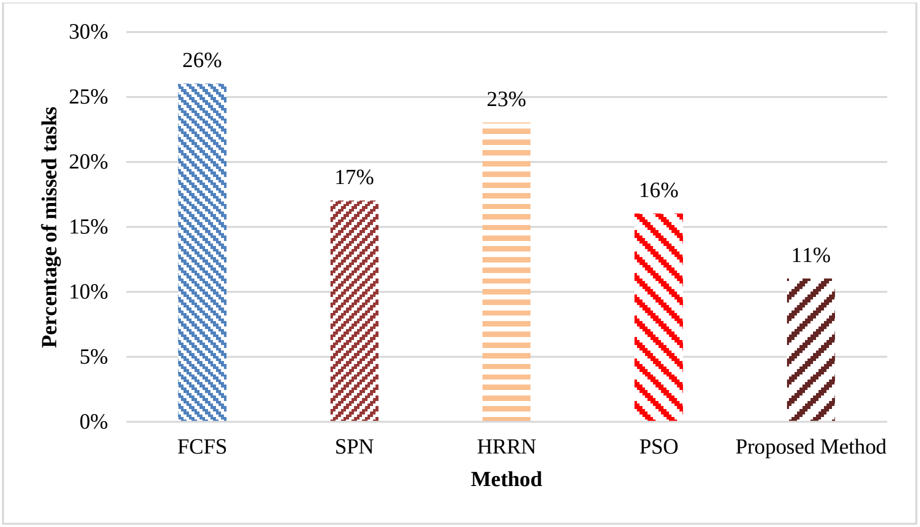

Fig. 5. Number of missed tasks comparison.

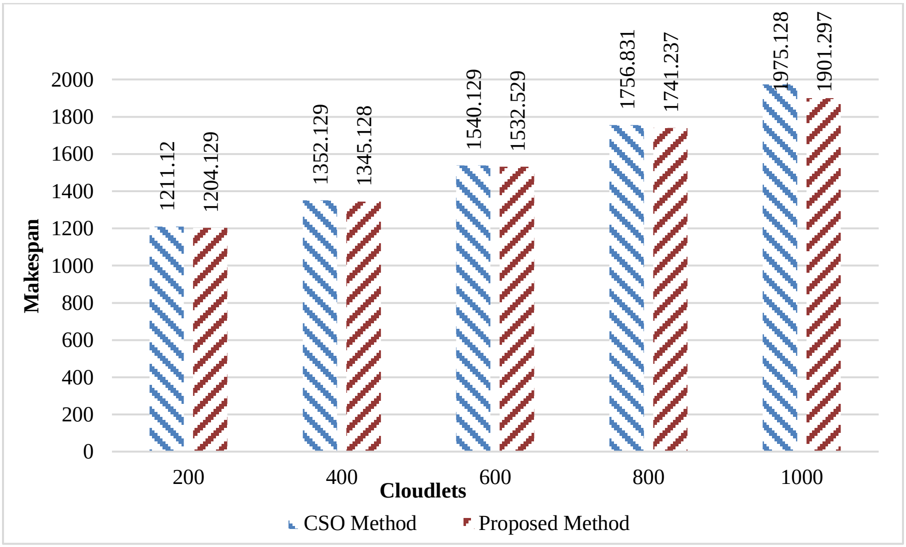

Fig. 6. Makespan comparison based on HP2CN workload.

## B. *First Experiment: Evaluation of the Proposed Method Compared to [33]*

This experiment compares the proposed method with the method presented in [33] regarding makespan and power consumption. HPC2N and NASA workloads were evaluated using 500 physical machines, 200 virtual machines, and 100 to 1000 tasks. As shown in Fig. 6 to 9, our method provides better performance. The proposed scheduler outperforms the existing CSO algorithm by approximately 10% in terms of makespan under HPC2N workloads. When compared to the CSO algorithm, our algorithm improved the makespan by 8% under NASA workloads. Energy consumption is an important parameter that impacts both the cloud provider and the cloud user. In the cloud computing paradigm, the goal is to minimize energy consumption by which cloud providers can effectively run tasks on virtual resources in the cloud by consuming a minimal amount of energy. The cloud user is also benefited from the availability of resources at a lower cost since resources are readily available. HPC2N and NASA workload archives were used to evaluate energy consumption. We compared our algorithm with the CSO algorithm under HPC2N workloads, and the results indicated a significant reduction in energy consumption, up to 10%. Compared to the CSO algorithm, our algorithm consumes up to 12% less energy under NASA workload archives.

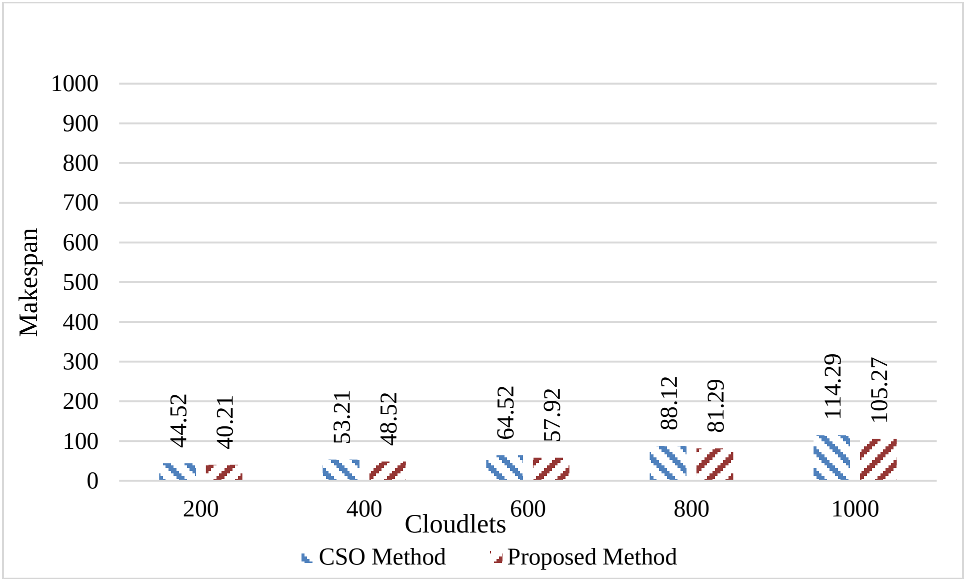

Fig. 7. Makespan comparison based on NASA workload.

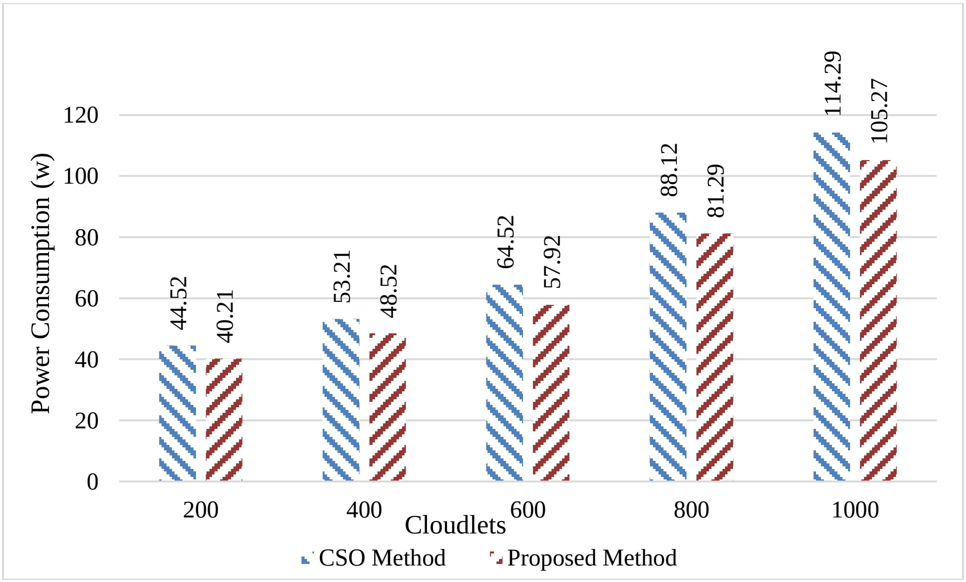

Fig. 8. Power consumption comparison based on HP2CN workload.

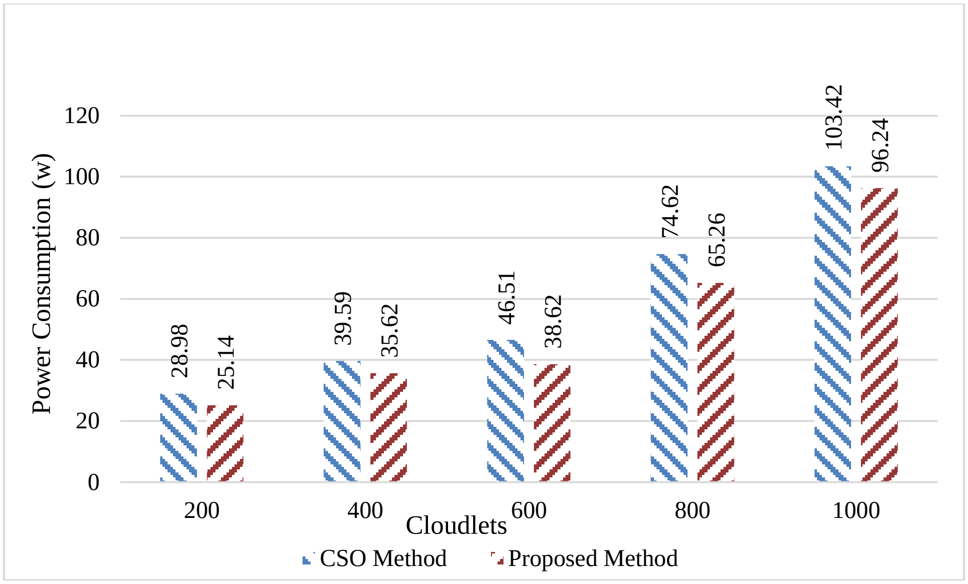

Fig. 9. Power consumption comparison based on NASA workload.

## V. Conclusion

Cloud computing allows users to access shared computing resources. Requests and demands are the basis of cloud computing, where users request resources from service providers and access them. Due to the dynamic nature of cloud environments and user requests changing over time, we need a scheduling method that can handle more tasks in less time. This paper proposed a new deadline-aware task-scheduling technique based on the firefly optimization algorithm. The experiments were conducted on different workloads. The experimental results proved that the proposed algorithm performed better than previous works concerning waiting time, execution time, missed tasks percentage, and resource utilization. Future work may examine the use of different heuristics to determine optimal initial conditions for FOA or other meta-heuristic algorithms. Our future plans include further exploring the time and space complexity of the proposed algorithm, examining the effective combination of artificial intelligence technology and task scheduling algorithm, and analyzing the energy consumption optimization of green cloud computing data centers.

## References

[1] A. Mehbodniya, J. L. Webber, R. Neware, F. Arslan, R. V. Pamba, and M. Shabaz, "Modified Lamport Merkle Digital Signature blockchain framework for authentication of internet of things healthcare data," Expert Systems, vol. 39, no. 10, p. e12978, 2022.

[2] F. Kamalov, B. Pourghebleh, M. Gheisari, Y. Liu, and S. Moussa, "Internet of Medical Things Privacy and Security: Challenges, Solutions, and Future Trends from a New Perspective," Sustainability, vol. 15, no. 4, p. 3317, 2023.

[3] S. P. Rajput et al., "Using machine learning architecture to optimize and model the treatment process for saline water level analysis," Journal of Water Reuse and Desalination, 2022.

[4] J. Akhavan, J. Lyu, and S. Manoochehri, "A deep learning solution for real-time quality assessment and control in additive manufacturing using point cloud data," Journal of Intelligent Manufacturing, pp. 1-18, 2023.

[5] R. N. Jacob, "Non-performing Asset Analysis Using Machine Learning," in ICT Systems and Sustainability: Proceedings of ICT4SD 2020, Volume 1, 2021: Springer, pp. 11-18.

[6] C. Han and X. Fu, "Challenge and Opportunity: Deep Learning-Based Stock Price Prediction by Using Bi-Directional LSTM Model," Frontiers in Business, Economics and Management, vol. 8, no. 2, pp. 51-54, 2023.

[7] S. H. Haghshenas, M. A. Hasnat, and M. Naeini, "A Temporal Graph Neural Network for Cyber Attack Detection and Localization in Smart Grids," arXiv preprint arXiv:2212.03390, 2022.

[8] S. Meisami, M. Beheshti-Atashgah, and M. R. Aref, "Using Blockchain to Achieve Decentralized Privacy In IoT Healthcare," arXiv preprint arXiv:2109.14812, 2021.

[9] S. Vairachilai, A. Bostani, A. Mehbodniya, J. L. Webber, O. Hemakesavulu, and P. Vijayakumar, "Body Sensor 5 G Networks Utilising Deep Learning Architectures for Emotion Detection Based On EEG Signal Processing," Optik, p. 170469, 2022.

[10] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[11] M. Mohseni, F. Amirghafouri, and B. Pourghebleh, "CEDAR: A cluster-based energy-aware data aggregation routing protocol in the internet of things using capuchin search algorithm and fuzzy logic," Peer-to-Peer Networking and Applications, pp. 1-21, 2022.

[12] F. Nzanywayingoma and Y. Yang, "Efficient resource management techniques in cloud computing environment: a review and discussion," International Journal of Computers and Applications, vol. 41, no. 3, pp. 165-182, 2019.

[13] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single-objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[14] P. Loubière and L. Tomassetti, "Towards cloud computing," TORUS 1–Toward an Open Resource Using Services: Cloud Computing for Environmental Data, pp. 179-189, 2020.

[15] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, A. Abraham, and N. M. Dankolo, "Cloud customers service selection scheme based on improved conventional cat swarm optimization," Neural Computing and Applications, pp. 1-22, 2020.

[16] S. K. Panda and P. K. Jana, "An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems," Cluster Computing, vol. 22, no. 2, pp. 509-527, 2019.

[17] V. Kunwar, N. Agarwal, A. Rana, and J. Pandey, "Load balancing in cloud—A systematic review," Big Data Analytics, pp. 583-593, 2018.

[18] Y. Wang, J. Wen, Q. Wu, L. Guo, and B. Tao, "A dynamic cloud service selection model based on trust and SLA in cloud computing,"

International Journal of Grid and Utility Computing, vol. 10, no. 4, pp. 334-343, 2019.

[19] P. Kumar and A. Verma, "Scheduling using improved genetic algorithm in cloud computing for independent tasks," in Proceedings of the international conference on advances in computing, communications and informatics, 2012, pp. 137-142.

[20] I. Attiya, M. Abd Elaziz, L. Abualigah, T. N. Nguyen, and A. A. Abd El-Latif, "An improved hybrid swarm intelligence for scheduling iot application tasks in the cloud," IEEE Transactions on Industrial Informatics, 2022.

[21] H. Yan, X. Zhu, H. Chen, H. Guo, W. Zhou, and W. Bao, "DEFT: Dynamic fault-tolerant elastic scheduling for tasks with uncertain runtime in cloud," Information Sciences, vol. 477, pp. 30-46, 2019.

[22] A. Amini Motlagh, A. Movaghar, and A. M. Rahmani, "Task scheduling mechanisms in cloud computing: A systematic review," International Journal of Communication Systems, vol. 33, no. 6, p. e4302, 2020.

[23] M. Soualhia, F. Khomh, and S. Tahar, "Task scheduling in big data platforms: a systematic literature review," Journal of Systems and Software, vol. 134, pp. 170-189, 2017.

[24] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," International Journal of Communication Systems, vol. 33, no. 16, p. e4583, 2020.

[25] A. Keivani, F. Ghayoor, and J.-R. Tapamo, "A review of recent methods of task scheduling in cloud computing," in 2018 19th IEEE Mediterranean Electrotechnical Conference (MELECON), 2018: IEEE, pp. 104-109.

[26] Y. Yu and Y. Su, "Cloud task scheduling algorithm based on three queues and dynamic priority," in 2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), 2019: IEEE, pp. 278-282.

[27] Y. Sun, J. Li, X. Fu, H. Wang, and H. Li, "Application research based on improved genetic algorithm in cloud task scheduling," Journal of Intelligent & Fuzzy Systems, vol. 38, no. 1, pp. 239-246, 2020.

[28] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, "A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing," IEEE Access, vol. 7, pp. 160916-160926, 2019.

[29] K. Prasanna Kumar and K. Kousalya, "Amelioration of task scheduling in cloud computing using crow search algorithm," Neural Computing and Applications, vol. 32, no. 10, pp. 5901-5907, 2020.

[30] L. Na, L. Fei, and D. W. Chao, "Cloud Task Scheduling Algorithm Based on Squid Operator and Nonlinear Inertia Weight," in 2019 IEEE Symposium Series on Computational Intelligence (SSCI), 2019: IEEE, pp. 3104-3109.

[31] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," ORSA journal on computing, vol. 6, no. 2, pp. 154-160, 1994.

[32] F. S. Milani and A. H. Navin, "Multi-objective task scheduling in the cloud computing based on the patrice swarm optimization," Int J Inf Technol Comput Sci, vol. 7, no. 5, pp. 61-66, 2015.

[33] S. Mangalampalli, S. K. Swain, and V. K. Mangalampalli, "Multi Objective Task Scheduling in Cloud Computing Using Cat Swarm Optimization Algorithm," Arabian Journal for Science and Engineering, vol. 47, no. 2, pp. 1821-1830, 2022.