Cloud Task Scheduling using the Squirrel Search Algorithm and Improved Genetic Algorithm

Qiuju DENG[1], Ning WANG[2], Yang LU[3]
Chongqing College of Mobile Communication, Chongqing, 401520, China[1, 2, 3]
Chongqing Key Laboratory of Public Big Data Security Technology, Chongqing, 401420, China[1]

***Abstract*—With cloud computing, resources can be networked globally and shared easily between users. A range of heterogeneous needs are met on demand by software, hardware, storage, and networking. Dynamic resource allocation and load distribution pose challenges for cloud servers. In this regard, task scheduling plays a significant role in enhancing the performance of cloud computing. With the increase in the number of users and the capability of cloud computing, cloud data centers are experiencing concerns regarding energy consumption. To leverage cloud resources energy efficiently and provide real-time services to users, a viable cloud task scheduling solution is required. To address these problems, this paper proposes a new hybrid task scheduling algorithm based on squirrel search and improved genetic algorithms for cloud environments. The proposed scheduling algorithm surpasses existing scheduling algorithms across multiple parameters, including makespan, energy consumption, and execution time.**

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

I. Introduction

Recent technological and scientific advances in Complementary Metal-Oxide Semiconductor (CMOS) [1, 2], machine learning [3], cloud computing [4], 5G connectivity [5, 6], Blockchain [7], artificial intelligence [8, 9], smart grids [10], Internet of Things (IoT) [11, 12], and optical networks [13, 14] are bringing numerous benefits to society. Schedulers (brokers) in cloud computing determine potential solutions for assigning constrained resources to requests in order to achieve multiple goals (e.g., energy consumption, response time, resource utilization, reliability) [15-17]. It is believed that the study conducted in [18] laid the foundation for modern scheduling techniques. Schedules are used in many applications today, including power system control, multi-media data object scheduling on the Internet, and manufacturing printed circuit boards [19]. Over the past three decades, distributed computing systems have become one of the most important aspects of modern scheduling [20]. In recent years, various standalone computers have been combined with working together as a cluster system. By integrating heterogeneous resources from geographically dispersed areas, grid systems overcome the shortcomings of cluster systems by using more resources [21]. Cloud computing has recently become popular, combining the strengths of clusters and grids [22].

Due to the wide solution space, most scheduling problems are NP-hard and require a long period of time to be resolved within a minimal period [23]. The scheduling of limited resources in modern computing systems cannot be optimized using a polynomial time-scheduling algorithm [24]. The researchers of [25] illustrated the dilemma posed in this case by giving a simple example: approximately 0.02 percent of the possible solutions consume up to 1.01 the necessary amount of time to reach the optimal result. It is proven that a complex problem can be extremely challenging to solve. Therefore, researchers have been motivated to develop effective algorithms to solve such scheduling problems. Scheduling techniques can be static or dynamic [26]. Due to the dynamic nature of cloud environments, more dynamic algorithms must be incorporated to achieve breathtaking results. In contrast, static algorithms are only used when workloads vary only slightly. Thus, deterministic methods cannot solve the task scheduling problem. This problem has been solved significantly in polynomial time by meta-heuristic algorithms, which are non-deterministic methods [27].

Virtualization technology and dynamic task scheduling techniques can benefit cloud service providers and users. By scheduling tasks effectively, resources are conserved (the resource utilization ratio is increased), and incoming tasks are also completed in the shortest possible time (the makespan is minimized) [28]. With the growing workloads in cloud data centers, task scheduling has become increasingly critical due to the scarcity of resources. In order to improve QoS criteria and the mapping of incoming tasks to available resources, cloud task scheduling needs further study. In scheduling, the goal is to determine optimal resources for executing incoming tasks, thereby enabling a scheduling algorithm to enhance various QoS factors such as response time, energy consumption, resource utilization, and makespan [29]. The rest of the paper is organized as follows. The next section reviews the previous works. Section III describes the proposed method. Experimental results are reported in Section IV. The conclusion is provided in Section V.

II. Related Work

A QoS-aware cloud task scheduling algorithm was proposed by Wu, et al. [30]. In the proposed algorithm, tasks are first prioritized using their special attributes, then sorted according to their priority. Second, the algorithm schedules tasks based on the sorted task queue according to the completion time for each task on different services. Based on CloudSim experiments, the algorithm can achieve good load balancing and performance by using priority and completion time to determine QoS. An improved sunflower optimization algorithm was introduced by Emami [31] for optimizing existing task scheduling algorithms. The algorithm schedules

tasks in polynomial time. Experimental results have shown that the algorithm outperforms its competitors. Makespan and energy consumption have improved by 0.74% and 3%, respectively, compared to the best counterpart.

Yang, et al. [32] developed a simplified cloud computing task scheduling model. This paper uses game theory to simplify cloud computing task scheduling algorithms as opposed to previous studies. This algorithm considers the reliability of a balanced task when scheduling tasks with game theory. A task scheduling model for computing nodes is developed based on the balanced scheduling algorithm. The rate allocation strategy is calculated using game strategy in the cooperative game model. Experimental results indicate that the proposed algorithm performs better than others.

Srichandan, Kumar [6] developed an approach to task scheduling that combined the advantages of two widely used biologically-inspired algorithms: genetic and bacterial foraging. This article makes two main contributions. In the first place, the scheduling algorithm minimizes the time between tasks, and in the second place, it reduces energy consumption economically and environmentally. According to experimental results, the proposed algorithm provides superior performance for convergence, stability, and solution diversity.

Abd Elaziz, et al. [33] presented a method for scheduling cloud tasks to minimize the time consumed scheduling different tasks across different virtual machines. This method uses Differential Evolution (DE) to improve the Moth Search Algorithm (MSA). The MSA mimics moth movements in nature using Levy flights and phototaxis as indicators of the ability to explore and exploit resources. The exploitation ability still needs to be improved so that DE can be used for local searches. Three experimental series are conducted to evaluate the proposed algorithm. An analysis of twenty global optimization problems is carried out using the traditional MSA and the proposed method in the first experiment. The proposed algorithm was compared to other meta-heuristic and heuristic algorithms on synthetic and real trace data in the second and third experimental series. Performance measurements in both experiments demonstrate that the proposed algorithm outperforms competitors.

Using cat swarm optimization algorithm, Mangalampalli, et al. [34] addressed data center-specific parameters, such as power consumption, migration time, and makespan. VM mapping was performed by calculating the priorities of tasks at the task level. Based on the cloudsim simulator, this algorithm generates random inputs for total power costs. HPC2N and NASA workload archives are used as inputs to the algorithm. The proposed algorithm is compared to existing algorithms such as PSO and CS. Using HPC2N and NASA workloads, significant improvements are observed in different parameters.

Various meta-heuristic algorithms have been used in the works discussed above. These approaches share the common characteristic of using random population to initialize the metaheuristics and hybrid metaheuristics. The initial population has a significant impact on metaheuristic algorithms. Randomness is a fundamental requirement for avoiding local minimum traps. However, the algorithm convergence can be improved if some particles are assisted heuristically in selecting effective or near optimum starting points. The proposed algorithm utilizes heuristic algorithms for initializing the papulation in order to significantly improve the algorithm's performance.

III. Proposed Method

There are many scheduling algorithms to minimize the tasks' completion time in distributed systems. These types of scheduling systems find the most proper resources to assign to the tasks. Minimizing the tasks' completion time does not lead to minimizing each task's execution time. Task scheduling goals in cloud computing are to propose optimal scheduling of the tasks with load balancing guarantee and guarantee Quality of Service (QoS) criteria like response time, execution time, system throughput, cost, reliability, and availability. A new method is proposed for scheduling cloud tasks.

A. Formulating the Problem

The utilized method has four main parts, including the network information server, the network resource broker, the tasks, and the resources that act in the following manner. Users send requests to process tasks. The information about the task is embedded in the request, including the required CPU time for each task, the size of each task, and the total number of tasks. The network resource broker starts calculating the program parameters after the received message from the user. Moreover, the information server provides the resources information for the network resource broker. The proposed method will be used to select the input for processing the resource. The local update of the nodes is performed after assigning a task to a resource. The global update of the nodes is performed after executing a task by a reference. Fig. 1 shows the flowchart of the proposed algorithm.

The execution results are transmitted to the user. The fitness function is the function that receives a candidate solution for a problem as input and provides an output that determines a good amount of the solution. The key characteristic of the optimization algorithms is determining the fitness value of each solution. The algorithm tries to schedule K tasks to M virtual machines in each repetition. Virtual machines are optimally scheduled in accordance with their processing capacity, given by Eq. (1).

$$Capacityvm_j = Mipsvm_j \times PesNumvm_j \quad (1)$$

where $pesNumbervmj$ is the number of processors in the vmj virtual machine and MIPSVMJ is the number of million instructions per second of all processors on VMJ virtual machine. Task scheduling reduces the execution time of virtual machine tasks. The execution time is estimated by Eq. (2).

$$ExecutionTime_j = \frac{TaskLength_j}{Capacityvm_j} \quad (2)$$

where $TaskLengthj$ denotes the length of the jth request on the queue, and Capacityvmj refers to the processing capacity of the virtual machine on the jth location of the solution (J=1, 2, ..., K).

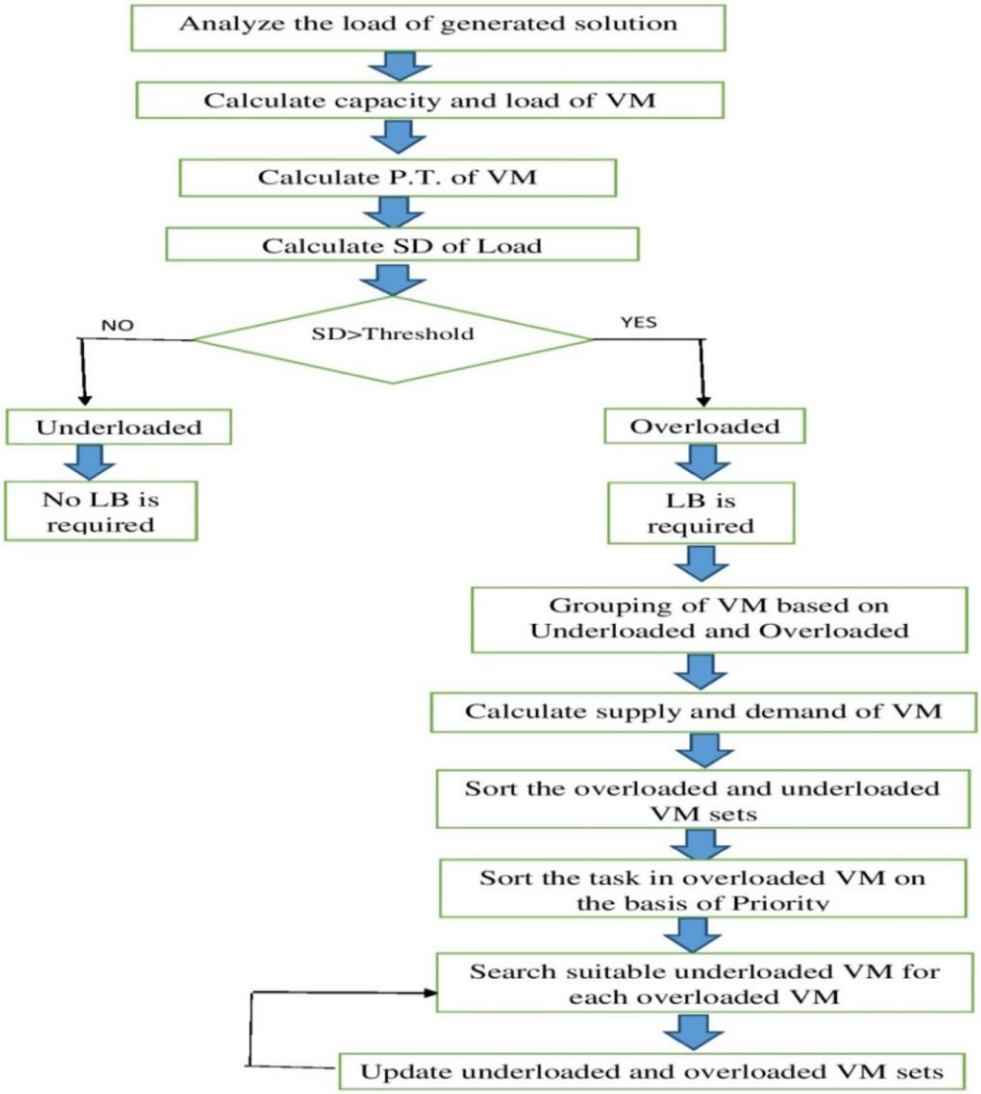

Fig. 1. Flowchart of the proposed algorithm.

The amount of load on the virtual machine and the amount of load resulting from accepting a new request are considered for load balancing in virtual machines. Hence the virtual machine load is defined as Eq. (3).

$$Loadvm_j = CurrentLoadvm_j + TaskLength_j \quad (3)$$

where $CurrentLoadVMj$ signifies the virtual machine load on the jth location of the solution (J=1, 2, ..., K). During task assignment, the standard deviation of the solution's virtual machine should be minimized for load balancing.

B. The Steps of the Proposed Method

Two main steps of the proposed method for task scheduling in this paper include:

- First step: GA to select the tasks and prioritize them for execution.
- Second step: using the Squirrel Search Algorithm (SSA) to map tasks to the virtual machines and their duration to reduce energy and fair load distribution.

1) Task selection for execution based on GA: First, in this section, general information is expressed about GA, and then the use of this algorithm to select the best task is explained. John Holland invented the main idea of GA from the evolutional theory of Darvin in 1967. Generally, GAs includes the following parts:

- Chromosome: Chromosomes in GAs show points in the search area and possible solutions to the considered problem. The number of genes (variables) on each chromosome (solution) is constant. Binary coding (binary strings) is used to present the chromosomes. A chromosome in this research shows a list of assigned resources for each task.
- Population: A population includes a set of chromosomes. A new population is generated with the same count of chromosomes using the impact of genetic operators on the population.

- Fitness function: First, a fitness function is provided to solve a problem using GAs. The fitness function in this research is based on the last task completion time duration, meaning that it is considered from the start time of the tasks to the last task completion in a parallel manner.
- Selection operator: This operator reproduces some chromosomes among the existing chromosomes in a population. Fitter chromosomes are more likely to reproduce. Elitist Selection is used in this research.
- Crossover operator: The crossover operator generates a new pair of chromosomes from a pair of chromosomes from the productive generation. Uniform crossover is used in this paper, and a random matrix is generated, namely a mask including 0 and 1 and the same length as the existing chromosomes. The mask chromosome determines which genes are transferred to the child from the first parent and which one from the second parent.
- Mutation operator: A mutation operator is applied to the chromosomes after crossover. This operator changes the content of a gene by randomly selecting an operator of a chromosome's gene.

Mapping the tasks of the application workflow to the distributed resources may have many objectives. The focus of this research is on minimizing the sum of the calculation time of the application workflow. The parallel workflow allows each task to have subtasks, and the subtasks are distributed among different resources in order to minimize the total completion time. so that each task can have some subtasks, and the subtasks are distributed among different resources to minimize the total time of the project completion. This system has two main parts:

- Task: it is the work performed in the cloud environment based on the user's request. Each task also is divided into some subtasks.
- Resource: each service in the cloud environment can assign one or some virtual machine and web services to each resource. These resources may have different processor powers and perform the service in different time durations and costs. A cloud computing system faces the challenge of selecting the resources for each task with the least amount of time and cost.

2) Machine selection by the SSA: Squirrel search is a memetic metaheuristic algorithm to find the optimal global solution via heuristic functions. This algorithm is based on memes evolution carried by interactive people and the global exchange of information among the population. In the SSA, the squirrels are transformed due to memetic evolution. In this algorithm, the squirrels are considered the hosts for the memes and are presented as a memetic vector. Each meme includes memo types showing a feature on the chromosome, like genes in GA. The squirrels can exchange their information and correct their memes. The amount of each squirrel search is adjusted by the memes improvement, and each squirrel's position is changed. SSA combines deterministic and random approaches. The deterministic approach makes it possible to use the response-level information efficiently to direct the heuristic search and the random components guarantee the flexibility and strength of the search.

The squirrel search is started with the primary population of P squirrels that are generated randomly from the problem area of Ω. In the Di-dimensional problems, the position of the i^{th} squirrel is presented as (xi1, xi2, …, xiD). Then the merit of each squirrel is calculated based on its position, and the squirrels are sorted decreasingly based on their merits. In the next step, the total population is divided into m groups. This division is performed so that each group includes n squirrels (P=m×n). During the division process, the first squirrel is located in the first group, the second one in the second group, the mth one in the mth group, and the (m+1)th one in the first group again. The squirrels with the best and the worst merit values are presented as Xb and Xw in each group, respectively. Moreover, the squirrels with the best merit among the population are presented as Xg. Then using an evolutional process, the worst existing squirrels' merit on each cycle of the algorithm is corrected.

3) Selecting the best machine by SSA: In this section, SSA is used to execute the tasks globally. In this method, each squirrel is considered a response to the problem, and the squirrels are distributed randomly. There are some sets with an equal number of squirrels. In order to assign tasks to virtual machines, three main measures should be considered, namely the task size, the machine processing power, and makespan.

The input tasks and the virtual machines are presented as $TaskLsit=\{t1,t2,\ldots,tn\}$ and $VM=\{vm1,vm2,\ldots,vmm\}$ respectively. The squirrel hybrid mutation evolutional approach maps the tasks to the local virtual machine. The algorithm steps are presented in the following.

4) Generating the First Generation: Like other evolutional algorithms, the primary population is generated randomly. In the proposed method, each virtual machine is considered a squirrel to perform the tasks. In each repetition of the algorithm, it tries to schedule K tasks by M virtual machines. The processing capacity of the virtual machines can affect the optimal scheduling of tasks to the virtual machines. Before assigning the squirrels to the sets, the fitting function value of each squirrel should be calculated using Eq. (4).

$$F = \frac{\text{Processing\_power}}{makespan} \quad (4)$$

This fitting function is calculated based on the machine processor and makespan. The lower the makespan, the better situation the machine has. Hence, the above equation results in the highest fitting function value for the most powerful machines.

After calculating the fitting function for all the squirrels' populations, they are sorted decreasingly, and there is a list of empty sets. The total population of the squirrels is divided into M sets. The division is performed so that each set has N squirrels. For the division, the first squirrel belongs to the first

set, the second one belongs to the second set, the Mth one belongs to the Mth set, and the (M+1)th belongs to the first set again. It is repeated until the last squirrel. Each M sets include N squirrels. Since the squirrels are sorted decreasingly based on the fitness function, the first and the last assigned squirrels to the set are the best and the worst solutions, respectively. Hence, the order of entering the squirrels into the sets is important. Locality and makespan criteria are considered to find the best answer by the squirrel algorithm, which are explained in the following. The processing capacity of each virtual machine is calculated using Eq. (5):

$$\text{Processing\_power} = \text{powerj} \times \text{PCount} \quad (5)$$

where power is the processing power of the virtual machine and Pcount is the number of empty processors. The execution time of each task on the virtual machine is estimated by Eq. (6):

$$\text{ExeTime} = \text{TaskTime} \times \text{Processing\_power} \quad (6)$$

where TaskTime is the size of the task which wants to be executed. The execution time of each virtual machine is different. Less execution time of a task on the virtual machine makes less makespan on the machine. In order to accomplish this, the following algorithm is applied.

The worst squirrel's location in each set of the local search based on the fitting function is improved according to the best answer location on that set or even the best answer of all sets. Hence the average of the squirrel fitting increases. The following algorithm is used for this aim:

- Step 1: the best and the worst squirrels of each set based on the fitting value are called $Xbest$ and $Xworst$, respectively.
- Step 2: the worst squirrel of each set ($Xworst$) tries to improve itself by exchanging its information rather than the best squirrel ($Xbest$). In order to reduce the makespan value achieved when all virtual machines are processing the same amount of data, the following improvement is performed.
- Step 3: two $Xbest$ and $Xw\ \ rst$ squirrels are selected so that their fitting function has the most difference, and this value should apply to all. Thus, the number of tasks of Xworst is transferred to Xbest. This transfer is performed until both fitting functions are equal.
- Step 4: after duplication of these two values, the list of squirrels in the set is sorted again. This process is performed for the next Xbest and Xworst.
- Step 5: this process is continued until the fitting function value of all squirrels is equal to the set fitting function average value.
- Step 6: all the sets are combined and sorted based on the fitting value, decreasingly, after internal evolution in each set. Then they are divided into some sets, and the evolution continues until the stop condition.

Usually, the stop condition of the algorithm is selected based on the constant variations of the best answer fitting or the algorithm repetition up to a determined number. In this problem, the considered stop condition is the determined value, $globalit$.

IV. Simulation

The proposed algorithm for task scheduling is implemented using Cloudsim. Moreover, the proposed method is simulated on the San Diego dataset. The San Diego dataset is a widely used benchmark dataset for task scheduling simulations. By using Cloudsim to simulate the proposed algorithm on the San Diego dataset, it allows researchers to compare their results with the existing literature on task scheduling and measure the performance of their proposed algorithm. This section compares the proposed method with [9] and [8] methods based on comparable criteria, including makespan, energy consumption, and execution time. This comparison allows for a clear assessment of the relative merits of the proposed method compared to the existing literature, highlighting the advantages in terms of performance and energy efficiency. Makespan determines the maximum time that each machine is active. If the distributions are not fair, this criterion is for different machines. It is the maximum time of the machine that works more than all other machines. The less general average of this criterion makes the better performance of the scheduling algorithm.

The proposed method is compared in the first experiment with the method presented in [35]. According to the results obtained in this experiment, the proposed method showed better results with regard to makespan, as shown in Fig. 1. This is because the proposed method is able to more efficiently distribute the tasks among the different machines, resulting in a lower makespan. Additionally, the proposed method is able to better account for the different capabilities of the machines, leading to a more even distribution of the tasks and better machine utilization. The proposed method is compared in the second experiment with the method presented in [34]. HPC2N and NASA workloads were used in this experiment to evaluate the proposed method. As shown in Fig. 2 and Fig. 3, the proposed method outperforms previous approaches regarding makespan time. The proposed method is able to better utilize the machines by accounting for the differences in the machines' capabilities. This means that it can better distribute the tasks to the machines, leading to a shorter makespan time, as seen in the results of the second experiment.

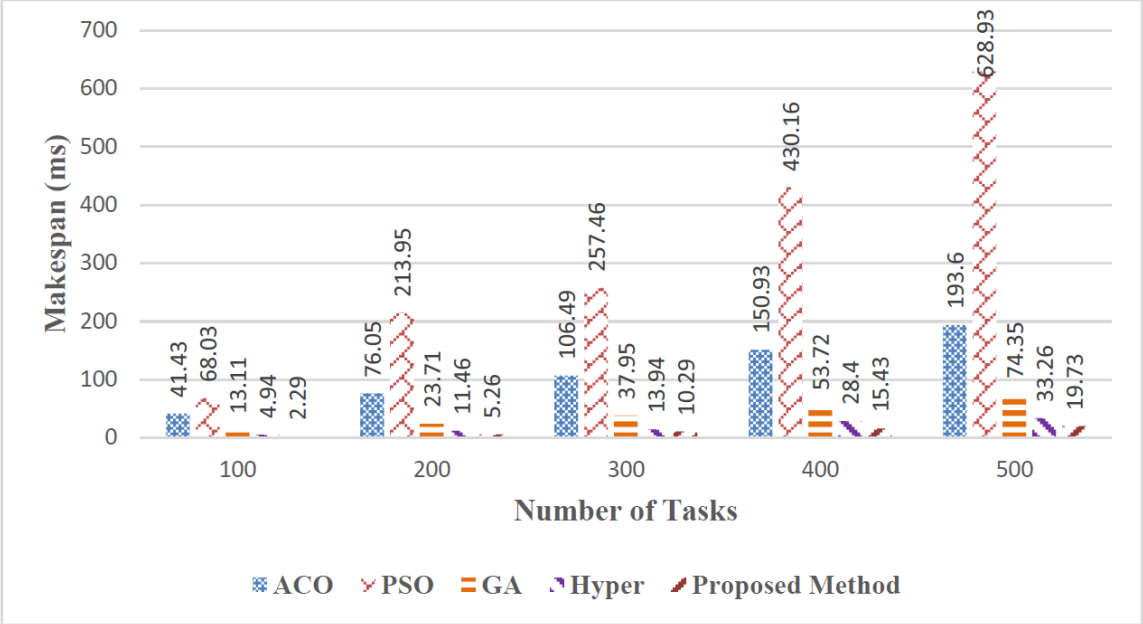

Fig. 2. Makespan comparison

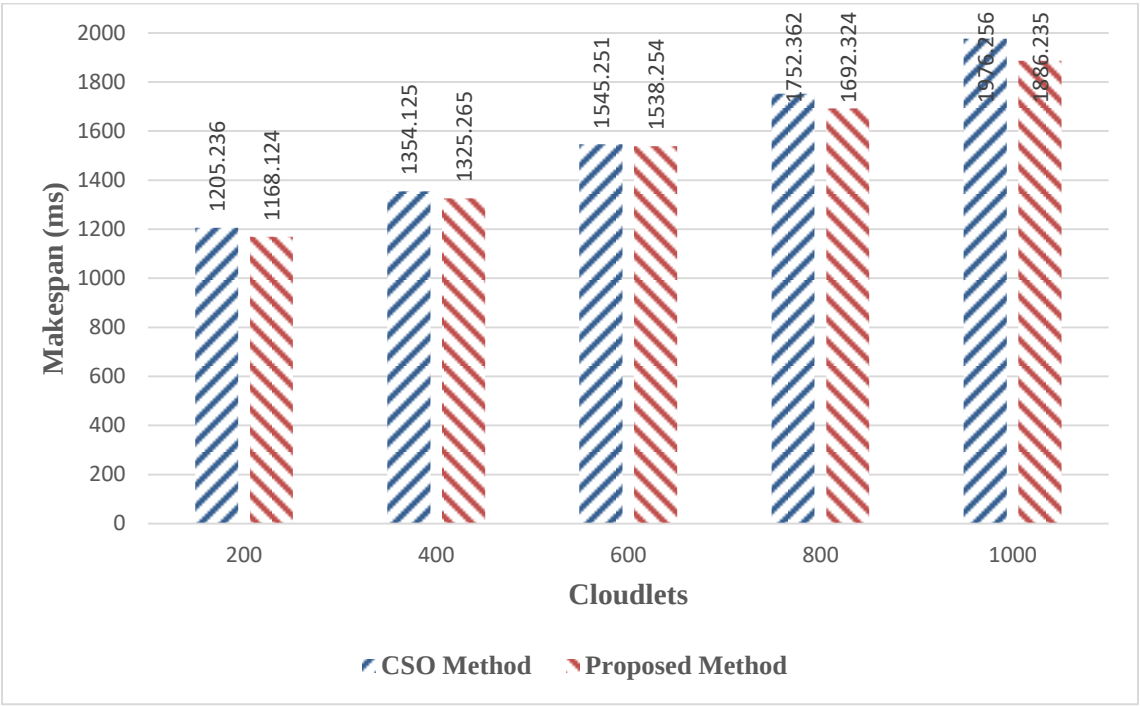

Fig. 3. Makespan comparison based on HP2CN workload.

Energy consumption criterion shows the amount of consumed energy for the execution of the tasks on the machines simulated in two different scenarios. In the first scenario, the number of machines is constant, and the number of tasks increases in each step. It is assumed that each task unit consumes one energy unit. In this scenario, the proposed method's performance is better. The cause of the increasing trend of the consumed energy diagram is the constant number of machines and the increasing number of tasks in each step. Execution time is the average time the tasks reach the resources. The less time, the better the scheduling algorithm. In the third experiment, the energy consumption and execution time of the proposed method is compared according to the data of the article [36]. This experiment uses five physical machines, 20 virtual machines, and 50 to 400 tasks. The obtained results are shown in Fig. 4. In the fourth experiment, the power consumption and execution time are evaluated based [37] dataset. Four physical machines and 5 - 50 virtual machines are used in this experiment. The obtained results are shown in Fig. 5 to 7.

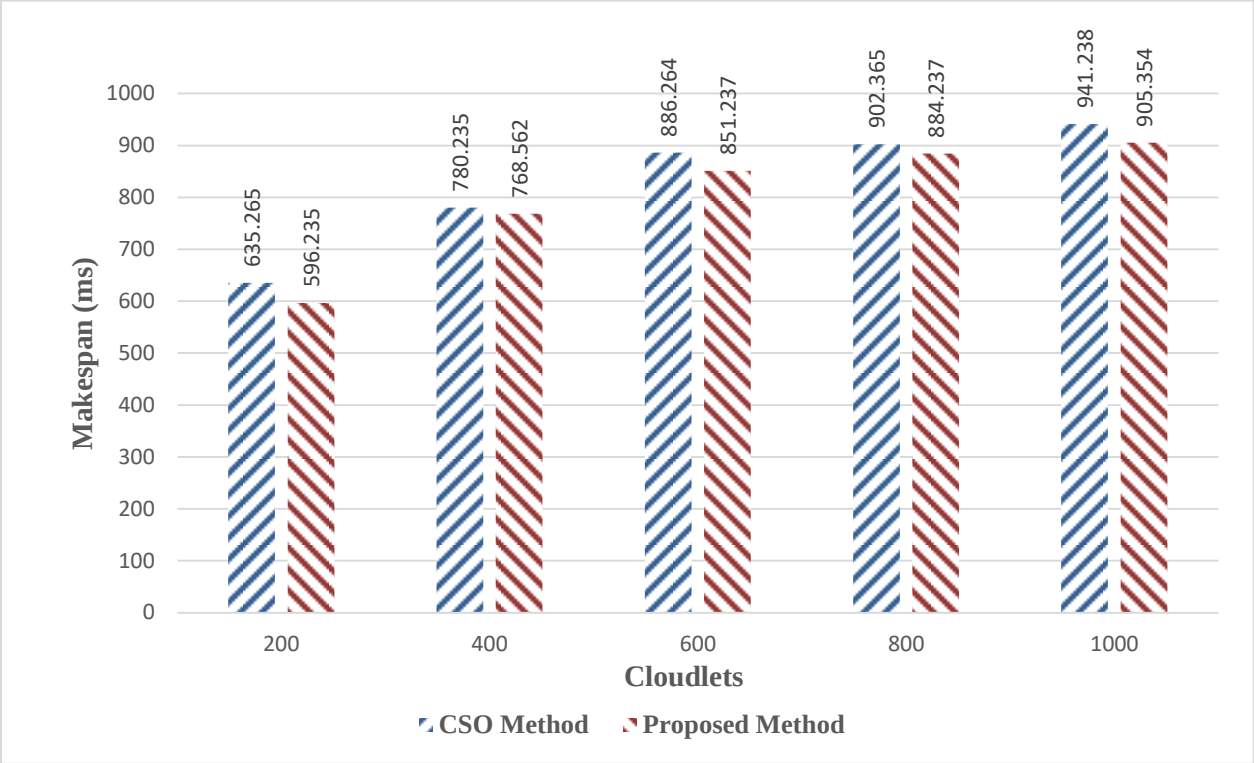

Fig. 4. Makespan comparison based on NASA workload.

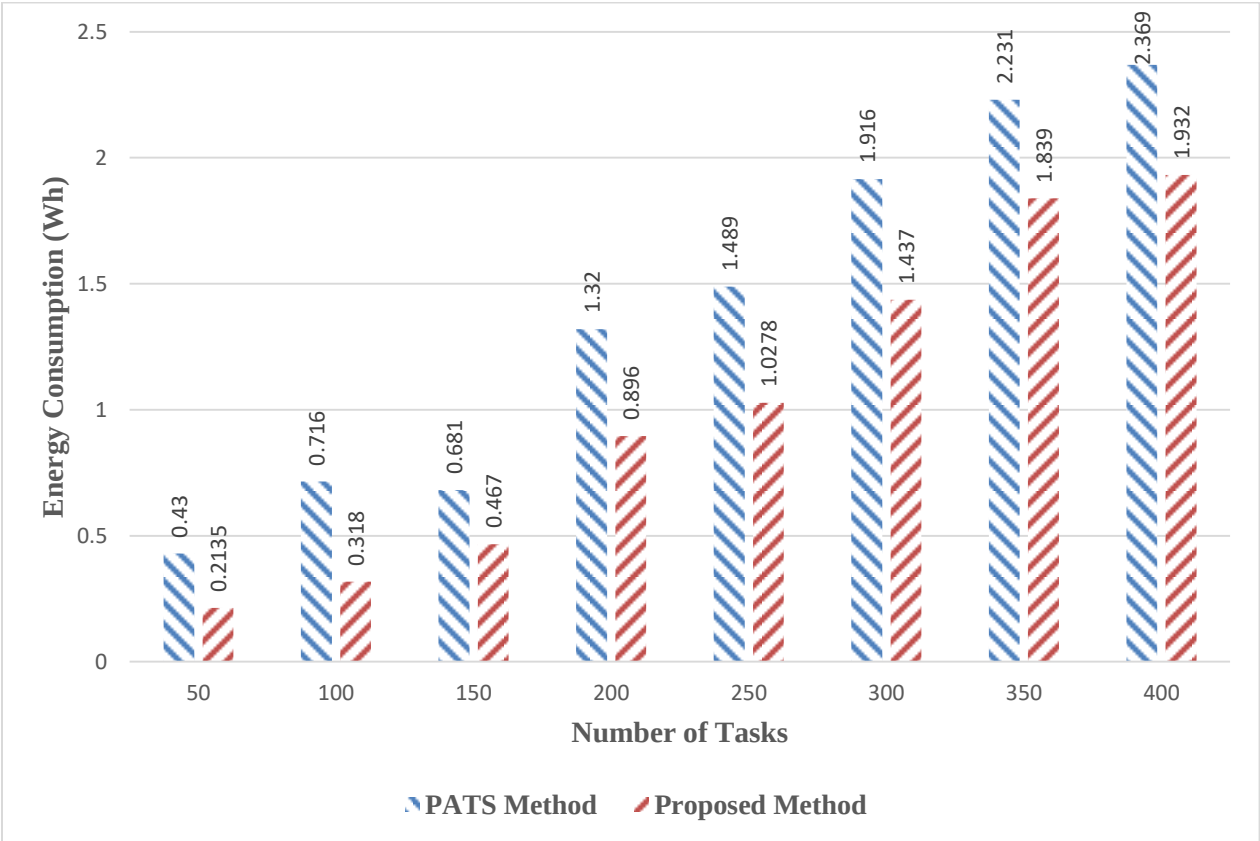

Fig. 5. Energy consumption comparison.

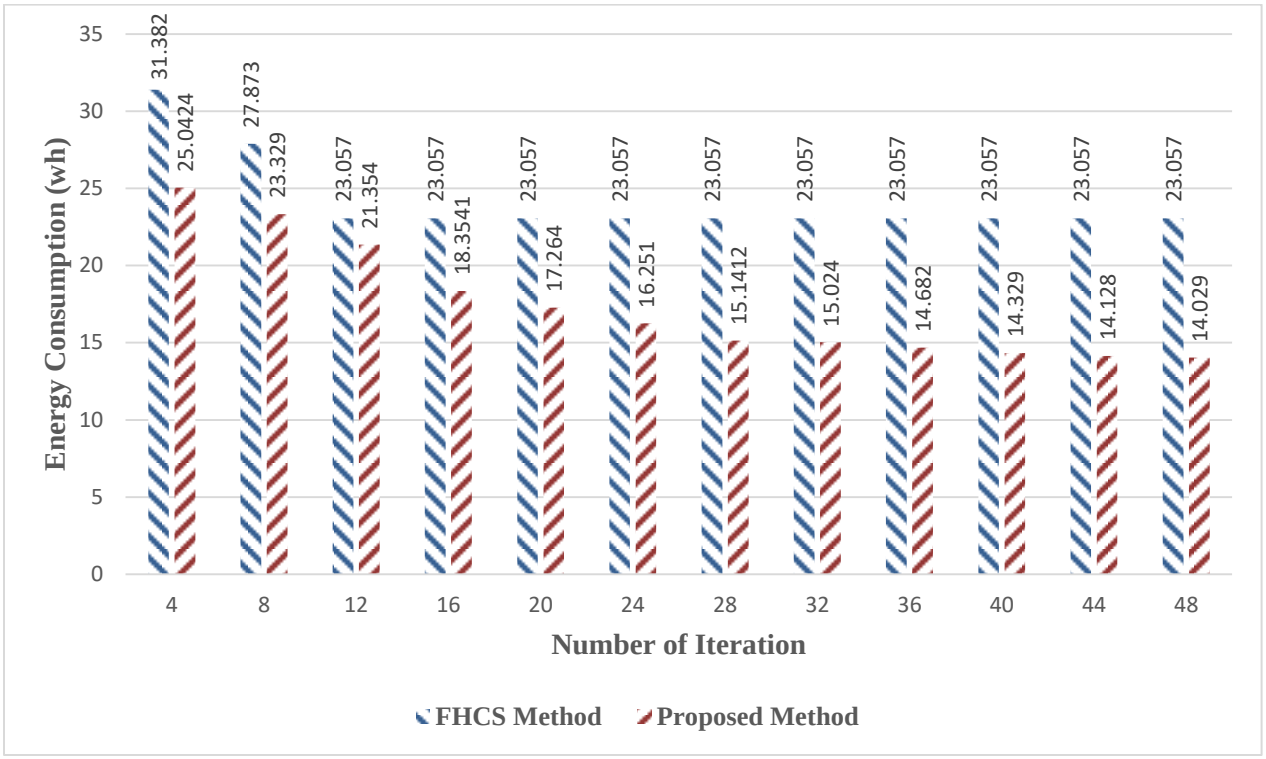

Fig. 6. Energy consumption comparison vs. iteration.

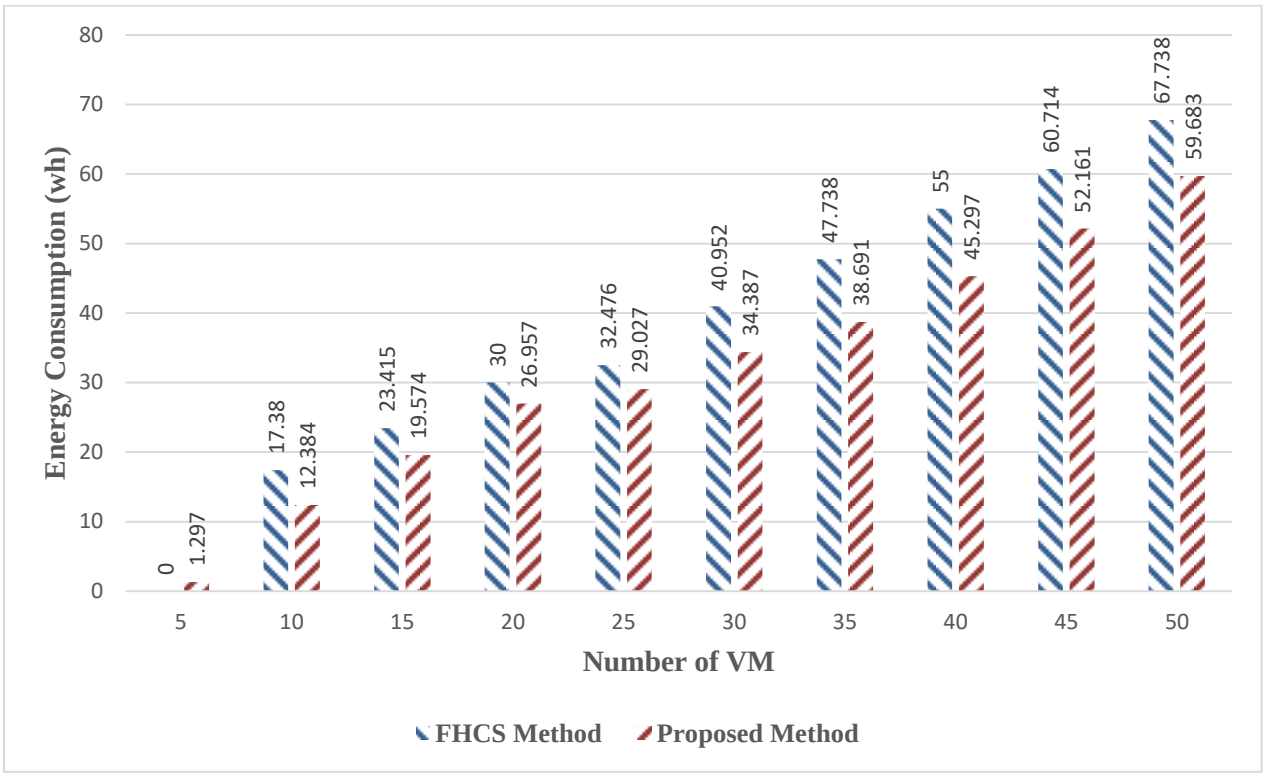

Fig. 7. Energy consumption comparison vs. number of VMs.

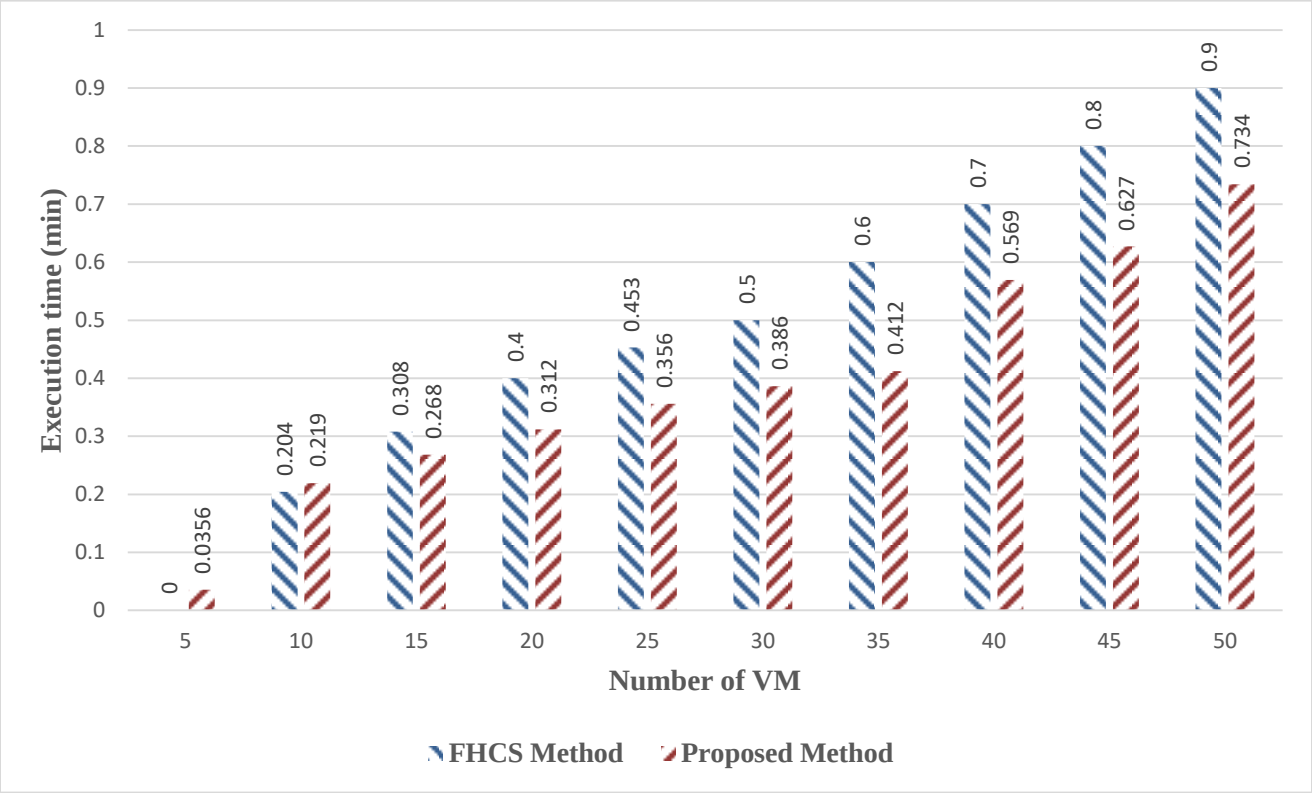

Fig. 8. Execution time comparison.

V. Conclusion

In this paper, by applying efficient scheduling to virtual machines, the efficiency of the system is enhanced, resulting in a shorter response time. It makes quick calculations and reduces energy consumption. This problem aims to apply an efficient scheduling method on virtual machines on a cloud system to meet all operational requests, and each performance criterion is optimized. Hence, metaheuristic algorithms are used. This algorithm is used by mathematical modeling of the political-social evolutional process to solve many optimization problems. This optimization evolutional strategy performance in convergence rate and reaching the global optimal is very high. While integrating multiple meta-heuristic methods may provide a hybrid heuristic with good performance, some meta-heuristics are not complementary, so combining them may not improve or even degrade performance. Performance is also affected by the integration strategy. In order to improve the performance of distributed systems on a wide range of aspects, we will study the complementarity of multiple meta-heuristics and develop an efficient integration strategy for the hybrid of multiple meta-heuristics.

References

[1] S. Seyedi and B. Pourghebleh, "A new design for 4-bit RCA using Quantum Cellular Automata Technology," Optical and Quantum Electronics, vol. 55, no. 1, p. 11, 2023.

[2] S. Seyedi, B. Pourghebleh, and N. Jafari Navimipour, "A new coplanar design of a 4-bit ripple carry adder based on quantum-dot cellular automata technology," IET Circuits, Devices & Systems, vol. 16, no. 1, pp. 64-70, 2022.

[3] J. Akhavan and S. Manoochehri, "Sensory data fusion using machine learning methods for in-situ defect registration in additive manufacturing: a review," in 2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2022: IEEE, pp. 1-10.

[4] T. Taami, S. Krug, and M. O'Nils, "Experimental characterization of latency in distributed iot systems with cloud fog offloading," in 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS), 2019: IEEE, pp. 1-4.

[5] P. He, N. Almasifar, A. Mehbodniya, D. Javaheri, and J. L. Webber, "Towards green smart cities using Internet of Things and optimization algorithms: A systematic and bibliometric review," Sustainable Computing: Informatics and Systems, vol. 36, p. 100822, 2022.

[6] I. Ataie, T. Taami, S. Azizi, M. Mainuddin, and D. Schwartz, "D 2 FO: Distributed Dynamic Offloading Mechanism for Time-Sensitive Tasks in Fog-Cloud IoT-based Systems," in 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), 2022: IEEE, pp. 360-366.

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

[8] F. Vahedifard, S. Hassani, A. Afrasiabi, and A. M. Esfe, "Artificial intelligence for radiomics; diagnostic biomarkers for neuro-oncology," World Journal of Advanced Research and Reviews, vol. 14, no. 3, pp. 304-310, 2022.

[9] S. A. Saeidi, F. Fallah, S. Barmaki, and H. Farbeh, "A novel neuromorphic processors realization of spiking deep reinforcement learning for portfolio management," in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022: IEEE, pp. 68-71.

[10] 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.

[11] 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.

[12] 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.

[13] F. Khosravi, M. Tarhani, S. Kurle, and M. Shadaram, "Implementation of an Elastic Reconfigurable Optical Add/Drop Multiplexer based on Subcarriers for Application in Optical Multichannel Networks," in 2022 International Conference on Electronics, Information, and Communication (ICEIC), 2022: IEEE, pp. 1-4.

[14] F. Khosravi, G. Mahdiraji, M. Mokhtar, A. Abas, and M. Mahdi, "Improving the performance of three level code division multiplexing using the optimization of signal level spacing," Optik, vol. 125, no. 18, pp. 5037-5040, 2014.

[15] 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.

[16] A. Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Multi-objective Task Scheduling in cloud-fog computing using goal programming approach," Cluster Computing, vol. 25, no. 1, pp. 141-165, 2022.

[17] B. Pourghebleh, N. Hekmati, Z. Davoudnia, and M. Sadeghi, "A roadmap towards energy-efficient data fusion methods in the Internet of Things," Concurrency and Computation: Practice and Experience, p. e6959, 2022.

[18] S. M. Johnson, "Optimal two-and three-stage production schedules with setup times included," Naval research logistics quarterly, vol. 1, no. 1, pp. 61-68, 1954.

[19] B. Sellami, A. Hakiri, S. B. Yahia, and P. Berthou, "Energy-aware task scheduling and offloading using deep reinforcement learning in SDN-enabled IoT network," Computer Networks, vol. 210, p. 108957, 2022.

[20] A. Kumar et al., "Optimal cluster head selection for energy efficient wireless sensor network using hybrid competitive swarm optimization and harmony search algorithm," Sustainable Energy Technologies and Assessments, vol. 52, p. 102243, 2022.

[21] 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.

[22] W. Shu, K. Cai, and N. N. Xiong, "Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing," Future Generation Computer Systems, vol. 124, pp. 12-20, 2021.

[23] 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.

[24] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," Cluster Computing, pp. 1-19, 2020.

[25] E. Taillard, "Some efficient heuristic methods for the flow shop sequencing problem," European journal of Operational research, vol. 47, no. 1, pp. 65-74, 1990.

[26] X. Chen et al., "A WOA-based optimization approach for task scheduling in cloud computing systems," IEEE Systems Journal, vol. 14, no. 3, pp. 3117-3128, 2020.

[27] 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.

[28] M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, "Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review," Journal of Grid Computing, pp. 1-30, 2020.

[29] 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.

[30] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. Zhou, "A task scheduling algorithm based on QoS-driven in cloud computing," Procedia Computer Science, vol. 17, pp. 1162-1169, 2013.

[31] H. Emami, "Cloud task scheduling using enhanced sunflower optimization algorithm," ICT Express, vol. 8, no. 1, pp. 97-100, 2022.

[32] J. Yang, B. Jiang, Z. Lv, and K.-K. R. Choo, "A task scheduling algorithm considering game theory designed for energy management in cloud computing," Future Generation computer systems, vol. 105, pp. 985-992, 2020.

[33] M. Abd Elaziz, S. Xiong, K. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," Knowledge-Based Systems, vol. 169, pp. 39-52, 2019.

[34] 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.

[35] A. Gupta, H. S. Bhadauria, and A. Singh, "Load balancing based hyper heuristic algorithm for cloud task scheduling," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 6, pp. 5845-5852, 2021.

[36] H. Zhao, G. Qi, Q. Wang, J. Wang, P. Yang, and L. Qiao, "Energy-efficient task scheduling for heterogeneous cloud computing systems," in 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2019: IEEE, pp. 952-959.

[37] B. B. Naik, D. Singh, and A. B. Samaddar, "FHCS: Hybridised optimisation for virtual machine migration and task scheduling in cloud data center," IET Communications, vol. 14, no. 12, pp. 1942-1948, 2020.