Energy Efficient Workflow Scheduling in Cloud Computing Systems using Particle Swarm Optimization

Abhishek Kumar[1], Santanu Ghosh[2], B Balaji Naik[3], and Pratyay Kuila[4]

Department of Computer Science and Engineering
National Institute of Technology Sikkim, South Sikkim-737139, India
m210001@nitsikkim.ac.in[1], santanu1109ghosh@gmail.com[2],
balajinaik07@gmail.com[3], pratyay_kuila@yahoo.com[4]

Abstract. This research paper proposes a novel approach for minimizing makespan and energy consumption in workflow scheduling for cloud computing systems using particle swarm optimization (PSO). Workflow scheduling is a critical task in cloud computing, where multiple tasks are assigned to each available virtual machine to ensure the precedence constraint of the workflow application. However, traditional scheduling methods often lead to longer makespan and higher energy consumption, which can negatively impact the overall efficiency and sustainability of the cloud infrastructure. The proposed PSO-based approach optimizes the task allocation and scheduling process by leveraging the swarm intelligence of particles to search for the optimal solution. The PSO algorithm is adapted to consider both makespan and energy consumption as objective functions, allowing for a more comprehensive and balanced optimization approach. The proposed approach is evaluated using a workflow application, and the results show that it outperforms traditional scheduling algorithms such as HEFT in terms of makespan and energy consumption.

Keywords: Workflow Scheduling· PSO· HEFT· Makespan· Energy Consumption.

1 Introduction

A lot of interesting work is going on in the scientific application field, and these applications are in need of high computation, broader bandwidth, and large storage for running complex applications, analyzing huge datasets, performing modeling tasks, and running complex simulations. Researchers are moving to the cloud for these resources as the cloud satisfies all the requirements, like allowing scientists to access large amounts of on-demand computation power, store and analyze datasets for astronomy, bioinformatics, or climatic research for collaboration, and provide a platform for easily sharing and grouping to accelerate research [1, 2].

In general, a scientific application generates a workflow in the form of a directed acyclic graph (DAG), which consists of a huge number of dependent and independent tasks with computation times and communication time. One of the greatest obstacles for cloud computing is workflow scheduling such that the dependency between tasks in the workflow is maintained [3–6]. A brief sketch of the generation and processing of scientific workflow applications is shown in Fig 1. Task scheduling is the assignment and management of task execution in a cloud environment. This can include scheduling the execution of individual tasks, as well as coordinating and managing the dependencies and relationships between tasks. For heterogeneous systems on the cloud, this difficulty is considered an NP-complete problem [7], and considering multiple objectives like makespan, energy, cost, reliability, throughput, and efficiency for heterogeneous systems is not trivial.

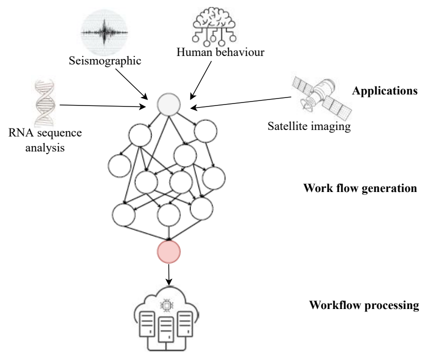


Fig. 1: An overview of generation of WAs

The motivation behind the minimization of makespan and energy consumption in cloud computing is to improve the efficiency and sustainability of the cloud computing infrastructure. Makespan refers to the time taken to complete a set of tasks in a computing system. In cloud computing, minimizing the makespan means reducing the time taken to complete a set of tasks, which can lead to faster and more efficient processing of data [8–11]. This is important for cloud computing applications such as real-time data processing and analysis, where a shorter makespan can result in quicker decision-making and better business outcomes.

On the other hand, energy consumption is a critical issue in cloud computing due to the massive amounts of energy required to power and cool the data centers that host the cloud infrastructure. Minimizing energy consumption in cloud computing can help reduce the environmental impact of cloud computing and lower the operational costs of running data centers. Additionally, reducing energy consumption can also lead to increased reliability and availability of the cloud computing infrastructure, as energy usage is closely tied to the heat generated by the data centers.

In summary, minimizing makespan and energy consumption in cloud computing can lead to faster processing times, reduced environmental impact, lower operational costs, and increased reliability and availability of the cloud infrastructure. Cloud providers are motivated to solve this problem, and much research is going on in this field. Most studies take into account only one or two objectives, such as single-objective [12], which minimizes only makespan, and multi-objective [1,3,5,9,13,14], which minimizes both cost and makespan.

The contributions of this paper are as:

– A novel workflow scheduling algorithm in a cloud computing environment using particle swarm optimization (PSO) is designed.
– The representation of the particle ensures that it always creates a full solution while respecting dependence constraints. The updating process at each iteration also ensures the precision of the updated particle locations.
– Using makespan and energy consumption, the designed fitness function is generated.

The remaining sections of this document are as follows: Section 2 contains related works. Section 3 concentrates on problem formulation and includes the system model, workflow application model, and terminology used to computer the objectives. An overview of PSO is shown in Section 4 with particle representation and updation. Section 6 has results and analysis, and the last section 7 has a set of conclusions and future directions.

2 Related Works

Usually, for solving workflow scheduling problems, most research uses either a heuristic or a meta-heuristic approach. This section is divided into two subsections: heuristic and meta-heuristic approaches.

A heuristic approach study often splits the research into stages. In the first phase, algorithms are used to rank the tasks in order of importance, and virtual machines are allocated to the tasks in the second phase depending on their importance [3, 5, 12, 13]. Topcuoglu et al. [12] introduced the heuristic known as the heterogeneous earliest completion time (HEFT) for process scheduling in diverse domains; it is still used as the foundation algorithm for new heuristic algorithms. It takes that into account, but it only minimizes makespan. Rizvi et al. [3] created the HBDCWS scheduling algorithm, which minimizes time and cost while meeting deadlines and budgets. This distributes the global budget to each task using a very simple budget distribution method. However, it is not energy-efficient. Fair budget-constrained workflow scheduling (FBCWS) by Rizvi et al. [13] minimizes delays when budget is a constraint. FBCWS functions in two stages: first, it categorizes then prioritizes the task, and then, in the second phase, VMs get chosen. This algorithm, however, considers only one QoS parameter at a time and either a deadline or a budget at a time. Zohu et al. [5] suggest a cloud environment's budget and deadline-constrained workflow scheduling (BDCWS). By taking the time frame and budget into account as limitations, the execution cost is kept to a minimum. However, the algorithm uses static virtual machine assignments.

When it comes to meta-heuristic approaches, most researchers optimize the output of heuristic algorithms [1, 9, 14]. Rizvi et al. [1] proposed the hybrid spider monkey optimization (HSMO) algorithm, which directs HSMO using a budget-deadline constrained algorithm. It reduces time and costs while staying within the deadline and budget. However, it failed to satisfy the QoS constraints. Aziza et al. [9] a hybrid genetic algorithm (HGA) was suggested that minimizes makespan within the time and budget constraints. It is a HEFT-GA hybrid that uses two-fold crossover and mutation to improve performance. However, it is not energy-effective. Middha et al. [14] propose a hybrid algorithm that combines particle search optimization (PSO) and PEFT to optimize the PEFT schedule result. It saves time, money, and energy while meeting a deadline and budget. However, its fitness function is not efficient.

3 System Models and Problem Formulation

3.1 System Model

Assume cloud computing consists of computing resources. These resources are allocated to cloud servers i.e., R=$\{CS_1, CS_2, \ldots CS_k\}$. Each cloud server consists of m number of virtual machines, i.e., CS=$\{VM_1, VM_2, \ldots VM_m\}$. There is one dedicated cloud manager who takes care of provisioning virtual machines on the cloud server. We also consider that all the VMs running on the cloud server can be heterogeneous in nature. Each cloud server consists of one resource manager, its responsibility is to store the information of VMs as well as for creation, allocation, and deletion of the VMs.

3.2 Workflow Application Model

Here, scheduling with dependency constraint or the tasks with a partial-order $\prec$ is considered. If tasks t_i and t_j, $t_i \prec t_j$ then task t_j cannot start execution on any machine until the task t_i finishes its execution. Also, $\prec$ is a transitive relation. Generally, this dependency is represented by a directed acyclic graph also called the task dependency graph (TDG), in which nodes represent individual tasks, and a directed edge from t_i to t_j represents the partial-order $\prec$ in the graph. Here there is a cost associated with every partial order $\prec$ there is a cost associated with the communication of data between two tasks. This cost can be represented by a communication matrix. Where $CT(i, j)$ will be the amount of communication delay which will be needed for the data to reach the task t_j from task t_i, also task t_j cannot start execution until it gets this data, and if both t_i and t_j are allocated on the same machine then the communication cost will be zero because the data will remain with the same machine. Multiple tasks can be depended on the same task in the TDG. The dependency between tasks is represented in the TDG, where edges are labeled with the communication costs. These types of problems are real-world problems and are used in various scientific and industrial applications. Let t_{size} be the size of a specific task t_i and p_j is the virtual machine's processing speed. The same is mentioned below in equation (1).

$$rt_{(i,j)} = \frac{t_{size}}{p_j} \tag{1}$$

RT represents the run time matrix, which consists of the runtime of different VMs on different tasks. $D_{(i,j)}$ is the length of data that has to be transferred from task t_i to t_j and $B_{(i,j)}$ be the communication channel bandwidth between two cloud servers. It is considered that $B_{(i,j)}$ is constant throughout the execution [15, 16]. The communication time depends on the size of the data and the bandwidth of the communication channel.

$$ct_{(i,j)} = \begin{cases} 0 & \text{if } v_i = v_j \\ \frac{D_{i,j}}{B} & \text{if } v_j \neq v_j \end{cases} \tag{2}$$

The RT and CT matrices are known at the beginning of the execution. However, RT and CT may be created through uniform distribution or can be acquired from real-world applications as log files [15–17]. During task execution, the computation speed is also considered constant.

3.3 Makespan

Before computing the makespan of the workflow application, it is required to compute the starting time of the tasks and the finishing time of the tasks. Starting time and Finishing time can be represented as $ST(t_i, v_j)$ and $FT(t_i, v_j)$ respectively. $ST(t_i, v_j)$ and $FT(t_i, v_j)$ computed as mentioned in the equation 3 and 4 respectively [18].

$$\mathrm{ST}\left(t_i, v_j\right) = \max\left\{\text{avail}[j], \max_{t_m \in \text{pred}(t_i)} \left(FT\left(t_m\right) + ct_{m,i}\right)\right\} \tag{3}$$

$$FT\left(t_i, v_j\right) = w_{i,j} + \mathrm{ST}\left(t_i, v_j\right) \tag{4}$$

The inner maximum block of the ST equation provides a time when all data required by t_i has been allocated to VM v_j. FT of task t_{exit} after all tasks in the graph are scheduled, the makespan will be the max of the finish time of the task t_{exit}.

$$\mathrm{M} = \max\left\{FT\left(t_{\text{exit}}\right)\right\} \tag{5}$$

3.4 Energy Consumption

The square of voltage supplied and frequency determines CMOS circuit energy usage [19–21]. Our system saves energy via dynamic voltage and frequency scaling, a potential energy-saving strategy. CMOS circuits use dynamic and static energy. Static energy consumption is ignored since dynamic power dissipation is the most expensive and time-consuming [22]. Dynamic power ($P_{dynamic}$) dissipation can be defined as (6).

$$P_{dynamic} = K \times V_j^2 \times f \tag{6}$$

Depending on the device's capabilities, K is a dynamic power constant. V_j is the voltage provide for the j^{th} VM and f is the frequency at the V_j of same VM. This allows us to define the energy consumption when computation is going on as (7).

$$E_{\text{busy}} = \sum_{j=1}^{m} K \times V_j^2 \times f \times t_j = \sum_{j=1}^{m} P_{dynamic} \times t_j \tag{7}$$

where t_j is the time for which task is running on VM v_j. f represents the VMs v_j frequency at V_j of voltage. When VM is not in use, the provided voltages and frequencies cannot be reduced to zero, thus the voltage must be at its lowest setting *lowestvoltage* i.e 70% of its capability, the equation for idle energy consumption is in (8).

$$E_{\text{idle}} = \sum_{j=1}^{m} K \times V_{lowest}^2 \times f_{lowest} \times t_{idle} = \sum_{j=1}^{m} P_{idle} \times t_{idle} \tag{8}$$

where V_{lowest} and f_{lowest} is the lowest voltage and frequency of the VM v_j at V_{lowest} respectively, t_{idle} represents the idle time of v_j. Form the equations (6), (8) and (7), the total energy consumption is calculated as (9):

$$E_{\text{total}} = E_{\text{busy}} + E_{\text{idle}} \tag{9}$$

3.5 Problem Formulation

The major difficulty is to allocate the n number of dependent tasks, $T = \{t_1, t_2, \ldots t_n\}$ to m number of virtual machines (VMs), i.e., $\{vm_1, vm_2, \ldots vm_m\}$ with the objective is to minimize the makespan and energy consumption. The fitness function is derived using the weighted sum method [23]. The entire range of the ω_k is from 0 to 1 and $\sum_0^2 \omega_k = 0$. Let a be β Boolean variable described as follows.

$$\beta_{(i,j)} = \begin{cases} 1, & \text{if } t_i \text{ is assigned to } vm_j \\ 0, & \text{else} \end{cases} \tag{10}$$

The problem can be stated as follows.

$$\text{Minimize } Fitness = \omega_1 \times \left(\frac{M}{Max_M} \right) + \omega_2 \times \left(\frac{E_{total}}{Max_{E_{total}}} \right) \tag{11}$$

Subject to

$$\sum_{j=1}^{m} \beta_{(i,j)} = 1, \forall i, 1 \leq i \leq n \tag{12}$$

$$\sum_{k=1}^{Obj} \omega_k = 1, 0 \leq \omega_k \leq 1, \forall k, 1 \leq k \leq Obj \tag{13}$$

Pre-emption is not allowed because of the Constraint (12). Constraint (13) ensures the summation of taken weight values is equal to one.

4 An Overview of PSO

Particle Swarm Optimization (PSO) is one of the swarm-inspired meta-heuristic optimization algorithms [24,25]. The population of the particle size is predefined. Every particle in the population gives a complete solution. The i^{th} particle of t^{th} iteration $(1 < t < M_{ite})$ can be represented as follows.

$$P_i^t = \{p_{(i,1)}^t, p_{(i,2)}^t, p_{(i,3)}^t \cdots p_{(i,d)}^t \cdots p_{(i,D)}^t\} \tag{14}$$

A particle, P_i^t has position value $p_{(i,d)}^t$ and velocity $v_{(i,d)}^t$. The fitness value is used to evaluate each particle to decide its quality. Each particle has its personal best, PB_i^t for each iteration. The population has the global best (GB^t) in each iteration. Particles are compared with their fitness value. At each iteration, the particle moves towards the destination position. The velocity is updated by using the following equations.

$$\begin{aligned} v_{(i,d)}^{(t+1)} = \omega \times v_{(i,d)}^{(t)} &+ c_1 \times r_1 \times [pb_{(i,d)}^{(t)} - p_{(i,d)}^{(t)}] \\ &+ c_2 \times r_2 \times [gb_d^{(t)} - p_{(i,d)}^{(t)}] \end{aligned} \tag{15}$$

where, inertia weight is ω, coefficients are c_1 and c_2. uniformly distributed random numbers are r_1 and r_2. After the computation of the velocity, the new position is updated as follows.

$$p_{(i,d)}^{(t+1)} = p_{(i,d)}^{(t)} + v_{(i,d)}^{(t+1)} \tag{16}$$

The fitness value is calculated after the new position of the particles has been successfully computed for a given iteration. Based on the fitness value, PB_i^t and GB^t are also updated. In order to arrive at a more optimal solution, the position and velocity of each and every particle are modified with each new iteration. The process of updating is carried out in a loop that continues until the maximum number of iterations has been reached. Finally, the global best is considered as the final solution.

5 Proposed Approach

5.1 Representation of Particles

We need to represent the particle in such a way that ensures it will always provide a comprehensive solution to the problem of scheduling the workflow. This can be fulfilled by ensuring that the dependency constraints are preserved. After we have successfully generated a valid sequence of the tasks that WAs will carry out, we will be able to schedule those tasks on the VMs. After the generation of a valid sequence, the particles are encoded according to the order in which the tasks were generated to be carried out. Consider N number of particle (P_i) $\forall i, 1 \leq i \leq N$ with position $(p_{(i,j)})$ $\forall j, 1 \leq j \leq D$. In this case, the number of sub-tasks for each WA is equal to the dimension. Each solution has position of $p_{(i,j)}$ that is randomly generated, $Rand[0, m]$, $0 < Rand[0, m] \leq m$. Representation of the particle shown in the Fig. 2. Initialization of the population is given in the Algorithm 1.

Algorithm 1 Population_Initialization

Input: No. of Particles(N), Dimension of Particle(D), Upper Bound(UB)
Output: A population.

```
$Population \leftarrow [\,]$
for $x = 1$ to $N$ do
    for $y = 1$ to $D$ do
        $Population[x][y] = random(UB, LB)$
    end for
end for
return Population
```

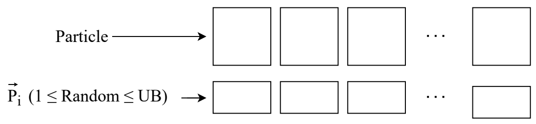


Fig. 2: Representation of a particle

In order to map the tasks to a VM, position $p_{(i,j)}$ is mapped to the virtual machine vm_{index}, where m represents the total number of VMs that are accessible from the cloud data center. Population initialization is written in the Algorithm 1.

5.2 Derivation of Fitness Function

The quality of a solution can be evaluated using the fitness function. The fitness function is derived from the objectives that have been mentioned earlier, such as the makespan and the energy consumption. The fitness of the particle P_i is computed as per eq. (11). The objective is to achieve the lowest possible fitness; in other words, a lower fitness indicates an improved particle or solution. The proposed PSO is shown in the Algorithm 2.

Algorithm 2 Particle Swarm Optimization (PSO)

Input: No. of Particles(N), Dimension(D), Max iteration(T)
Output: Global best

```
Population = Population_Initialization(N, D, UB)                ▷ Algorithm 1
Calculate $\boldsymbol{fit}$ using equation (11)
Find personal and Global best.
while z < T do
    for $v = 1$ to $N$ do
        Update $\boldsymbol{Velocity[i]}$ using equation (15)
        Update $\boldsymbol{Population[i]}$ using equation (16)
        Update $fit_{new}$ using equation (11)
        $fit \leftarrow [\ ]$
        if $fit_{new}[v] < fit[v]$ then
            $fit[v] = fit_{new}[v]$
            $G_{best} = min(fit)$
        end if
    end for
    $z = z + 1$
end while
Global Best = $G_{best}$
```

6 Result and Analysis

Here, we have considered the above given DAG (Fig 3) for executing the PSO and generating the best possible schedule. Table 1 represents the expected runtime of tasks on different VMs. According to HEFT algorithm [12] the sequence generated as follows (Task,VM) - [(0,2),(3,1), (2,2), (1,0), (4,2), (5,1), (8,1),(6,2), (7,0), (9,1)] and the makespan is 80. With the help of PSO, we improved the makespan and energy consumption over HEFT. We have executed PSO 10 times and computed makespan and energy consumption, and we have taken the average of all 10 results to compare with the HEFT algorithm, the comparison is in Table 2. Fig. 4 illustrates the results of an analysis of the system's performance in terms of its makespan and its energy consumption.

7 Conclusions

In this paper, we propose PSO to optimize the workflow scheduling in the cloud computing environment. The proposed algorithm evaluated a variety of criteria, including the makespan and the energy consumption. particle representation in such a way that it always preserves dependency constraints. At each iteration, a new position is restricted within the specified range. We have tested our approach on various workflow applications and also presented one of them in the paper. The method significantly outperforms HEFT in terms of makespan and energy consumption. In the future, we may consider various other objectives while considering their conflicting nature. Here we have considered only non-preemptive

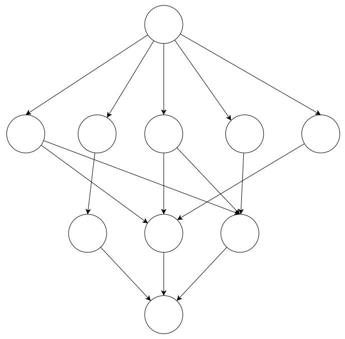


Fig. 3: Sample DAG with 10 tasks

Table 1: Expected runtime of tasks

$Tasks$	VM_1	VM_2	VM_3
0	14	16	9
1	13	19	18
2	11	13	19
3	13	18	7
4	12	13	10
5	13	16	9
6	7	15	11
7	5	11	14
8	18	12	20
9	21	7	16

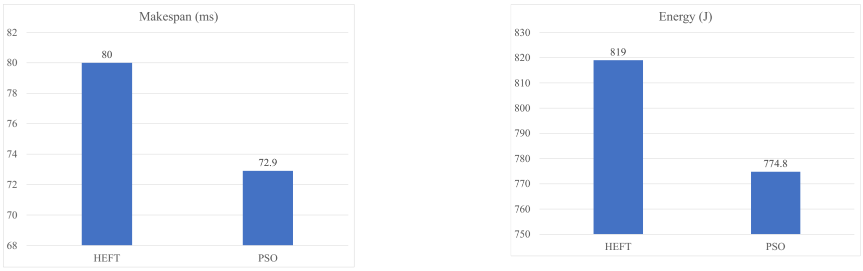
nature applications. In the future, pre-emptive applications with constraints like deadlines and budgets of the workflow application may be considered.

References

1. N. Rizvi, R. Dharavath, and D. R. Edla, "Cost and makespan aware workflow scheduling in iaas clouds using hybrid spider monkey optimization," *Simulation Modelling Practice and Theory*, vol. 110, p. 102328, 2021.
2. T. Biswas, P. Kuila, and A. K. Ray, "A novel scheduling with multi-criteria for high-performance computing systems: an improved genetic algorithm-based approach," *Engineering with Computers*, vol. 35, no. 4, pp. 1475–1490, 2019.

Table 2: Comparison of HEFT and PSO

Algorithm	**Makespan (ms)**	**Energy (J)**
HEFT	80	819.50
PSO	72.9	774.80



(a) Makespan (b) Energy

Fig. 4: Comparison of HEFT and PSO

3. N. Rizvi and D. Ramesh, “Hbdcws: heuristic-based budget and deadline constrained workflow scheduling approach for heterogeneous clouds,” *Soft Computing*, vol. 24, no. 24, pp. 18 971–18 990, 2020.
4. T. Biswas, P. Kuila, A. K. Ray, and M. Sarkar, “Gravitational search algorithm based novel workflow scheduling for heterogeneous computing systems,” *Simulation Modelling Practice and Theory*, vol. 96, p. 101932, 2019.
5. N. Zhou, W. Lin, W. Feng, F. Shi, and X. Pang, “Budget-deadline constrained approach for scientific workflows scheduling in a cloud environment,” *Cluster Computing*, pp. 1–15, 2020.
6. T. Biswas, P. Kuila, and A. K. Ray, “A novel resource aware scheduling with multi-criteria for heterogeneous computing systems,” *Engineering Science and Technology, an International Journal*, vol. 22, no. 2, pp. 646–655, 2019.
7. M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, “Towards workflow scheduling in cloud computing: a comprehensive analysis,” *Journal of Network and Computer Applications*, vol. 66, pp. 64–82, 2016.
8. T. Biswas, P. Kuila, and A. K. Ray, “A novel workflow scheduling with multi-criteria using particle swarm optimization for heterogeneous computing systems,” *Cluster Computing*, vol. 23, no. 4, pp. 3255–3271, 2020.
9. H. Aziza and S. Krichen, “A hybrid genetic algorithm for scientific workflow scheduling in cloud environment,” *Neural Computing and Applications*, vol. 32, no. 18, pp. 15 263–15 278, 2020.
10. A. S. Thakur, T. Biswas, and P. Kuila, “Binary quantum-inspired gravitational search algorithm-based multi-criteria scheduling for multi-processor computing systems,” *The Journal of Supercomputing*, vol. 77, pp. 796–817, 2021.
11. T. Biswas, P. Kuila, and A. K. Ray, “Multi-level queue for task scheduling in heterogeneous distributed computing system,” in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. IEEE, 2017, pp. 1–6.
12. H. Topcuoglu, S. Hariri, and M.-Y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.
13. N. Rizvi and D. Ramesh, “Fair budget constrained workflow scheduling approach for heterogeneous clouds,” *Cluster Computing*, vol. 23, no. 4, pp. 3185–3201, 2020.

14. K. Middha and A. Verma, "Peft-based hybrid pso for scheduling complex applications in IoT," in *Smart Computational Strategies: Theoretical and Practical Aspects*. Springer, 2019, pp. 273–286.
15. Y. C. Lee, H. Han, A. Y. Zomaya, and M. Yousif, "Resource-efficient workflow scheduling in clouds," *Knowledge-Based Systems*, vol. 80, pp. 153–162, 2015.
16. S. Su, J. Li, Q. Huang, X. Huang, K. Shuang, and J. Wang, "Cost-efficient task scheduling for executing large programs in the cloud," *Parallel Computing*, vol. 39, no. 4-5, pp. 177–188, 2013.
17. M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," *Future Generation Computer Systems*, vol. 48, pp. 1–18, 2015.
18. A. S. Thakur, T. Biswas, and P. Kuila, "Gravitational search algorithm based task scheduling for multi-processor systems," in *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN).* IEEE, 2018, pp. 253–257.
19. Y. Zhang, X. S. Hu, and D. Z. Chen, "Task scheduling and voltage selection for energy minimization," in *Proceedings of the 39th annual Design Automation Conference*, 2002, pp. 183–188.
20. N. B. Rizvandi, J. Taheri, and A. Y. Zomaya, "Some observations on optimal frequency selection in dvfs-based energy consumption minimization," *Journal of Parallel and Distributed Computing*, vol. 71, no. 8, pp. 1154–1164, 2011.
21. K. H. Kim, R. Buyya, and J. Kim, "Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters," in *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07).* IEEE, 2007, pp. 541–548.
22. V. Venkatachalam and M. Franz, "Power reduction techniques for microprocessor systems," *ACM Computing Surveys (CSUR)*, vol. 37, no. 3, pp. 195–237, 2005.
23. D. Chaudhary and B. Kumar, "Cloudy gsa for load scheduling in cloud computing," *Applied Soft Computing*, vol. 71, pp. 861–871, 2018.
24. P. Kuila and P. K. Jana, "Energy efficient clustering and routing algorithms for wireless sensor networks: Particle swarm optimization approach," *Engineering Applications of Artificial Intelligence*, vol. 33, pp. 127–140, 2014.
25. P. Maratha, K. Gupta, and P. Kuila, "Energy balanced, delay aware multi-path routing using particle swarm optimisation in wireless sensor networks," *International Journal of Sensor Networks*, vol. 35, no. 1, pp. 10–22, 2021.