Dicle University
Journal of Engineering
https://dergipark.org.tr/tr/pub/dumf
duje.dicle.edu.tr

Research Article

Hybrid Heuristic and Particle Swarm Optimization Approach to Cloud Task Scheduling

Cebrail BARUT[1]*, Kazım FIRILDAK[2]

[1] Firat University, Continuing Education Center, cbarut@firat.edu.tr, Orcid No: 000-0003-2756-5434

[2] Firat University, Department of Computer Technology, kfirildak@firat.edu.tr, Orcid No: 0000-0002-1958-3627

ARTICLE INFO

Article history:

Received 20 March 2025
Received in revised form 8 May 2025
Accepted 31 July 2025
Available online 30 September 2025

Keywords:

Cloud Computing, Task Scheduling, Heuristic Particle Swarm Optimization

Doi: 10.24012/dumf.1662221

* Corresponding author

ABSTRACT

Scheduling tasks on cloud systems is a critical optimization problem that aims to distribute tasks among available resources in the most effective way. This issue falls under the category of NP-hard problems and generating exact and deterministic solutions requires high computational costs. Metaheuristic approaches have proven to provide successful results in solving such problems. Particle Swarm Optimization (PSO), one of these algorithms, is a widely used method in the literature due to its advantages, such as fast convergence, simple applicability, and low computational cost. In this study, a hybrid heuristic-based Particle Swarm Optimization approach is proposed to improve task scheduling efficiency. The proposed approach improves the solution quality by integrating a heuristic mechanism into the random population generation process of PSO. In comparison to First Come First Serve, Ant Colony Optimization (ACO), and conventional PSO, the suggested approach delivers better makespan and reduced energy consumption, according to the simulation analysis carried out in the CloudSim simulation environment. According to simulations, Heuristic PSO outperforms traditional PSO and ACO methods in terms of makespan time, reducing it by an average of 61.42% and 62.84%, respectively. It also uses 26.18% less energy than PSO and 27.33% less than ACO, according to its energy consumption data. The results show that the suggested method offers a more effective substitute for scheduling tasks in cloud computing systems.

Introduction

The architecture of cloud services, which are becoming an essential part of the modern digital world, is extremely complicated. These systems can consist of a variety of components, ranging from network servers to personal computers, and are built by integrating numerous computers with advanced engineering techniques [1]. They can react quickly to different user task demands because of their large resource pool [2].

In terms of their vast hardware infrastructure and services, cloud systems are complicated, and managing them can be difficult [3]. Task scheduling is a crucial procedure in these systems that guarantees that tasks are assigned to the best available resources [4]. Task scheduling is made considerably more difficult by the cloud environment's abundance of large-scale resources. By guaranteeing effective resource utilization, an efficient task scheduling system maximizes workloads. By taking into account various user requests, it simultaneously enhances resource management, boosts performance, lowers expenses, and greatly improves Quality of Service (QoS). For this reason, the task scheduling procedure is crucial to the efficacy and durability of cloud systems. Because task scheduling in cloud systems is a complicated problem that needs to be optimized, numerous approaches have been put out in the literature to address this issue [5]. Metaheuristic approaches are widely employed in task scheduling problems because they can produce efficient solutions by conducting effective searches in a vast solution space. These techniques can, however, occasionally have drawbacks such becoming trapped in local minima, slow convergence, and high processing costs. The mixed usage of heuristic approaches and metaheuristics gives substantial advantages to overcome these issues and obtain more effective solutions. By employing strategies unique to the problem's structure, heuristic approaches can generate quick and effective first solutions; however, metaheuristic methods optimize these answers and arrive at the global optimum more quickly. This hybrid strategy can offer high efficiency in task scheduling processes and has the ability to lower computing costs while improving the quality of the solutions.

This work proposes a heuristic approach to a hybridized version of Particle Swarm Optimization (PSO) [6], a metaheuristic method commonly employed in cloud system task scheduling research. Below is a list of the suggested method's most inventive features:

• Heuristic approaches reduce the propensity of conventional PSO to become trapped in local minima, leading to quicker and more effective solutions.

• In certain applications, PSO's poor convergence speed is a drawback. The suggested approach strengthens with heuristic elements to achieve faster convergence.

• By improving both makespan (completion time) and total energy consumption, the suggested approach helps cloud systems become more sustainable.

• The addition of heuristic components to the PSO algorithm yields more reliable and superior results when compared to First Come First Server (FCFS), ACO, and PSO methods.

The remaining of the paper is organized as follows: Section 2 presents the literature review. Section 3 describes the problem definition and methodology. Section 4 presents and discusses the experimental results. Finally, Section 5 concludes the paper by highlighting the main findings.

Literature Research

Real-time computing capabilities can be unilaterally provided and managed by cloud computing without the need for human contact[7]. It uses task scheduling to perform these operations. Task scheduling is the procedure used in cloud systems [8]. Of allocating user tasks to the best virtual machines based on a predetermined plan. The cloud system's dynamic and multifaceted structure must be considered in the task scheduling method. The goal for which the task scheduling technique is employed is also crucial. Scheduling tasks on cloud systems can be designed to include one or more of these factors, including makespan, energy consumption, and financial cost. The task requested by the users can be completely allocated to the virtual machines or the tasks can be assigned to the servers in parts. In this case, it is necessary to consider the interdependencies of the parts within the task. In general, task scheduling methods used in cloud systems can be shown in Fig.1.

Figure 1. Task categories and the nature of the suggested.

Heuristic algorithms offer an effective solution for low-complexity and static systems [9]. The algorithms are easy to implement and have low computational cost. However, they give low performance to dynamic systems such as clouds and edges. Since heuristic methods are designed for a specific problem, they are difficult to apply directly to different problems. In the literature, Min-Min, Max-Min, Round Robin (RR), Shortest Job First (SJF), FCFS and Best Fit (BF) are the main heuristic methods. A modified round-robin (MRR) algorithm, an improved version of RR, is proposed [10]. The proposed method gave better results in terms of average waiting time and turnaround time. An improved Min-Min method that maximizes the total execution time and resource utilization of tasks is proposed [11]. Experimental results show that the proposed method achieves the best makespan values compared to the Sufferage and Min-Min algorithms.

Metaheuristic methods generally provide effective solutions for complex and dynamic task scheduling problems in cloud environments [12]. Numerous NP-hard problems can be effectively solved using metaheuristic techniques, which have also been successfully adapted for use in cloud systems. Metaheuristic algorithms used in the literature can be modified to address issues with task scheduling, with appropriate representation. PSO [13], Grey Wolf Algorithm (GWO) [14], Firefly Algorithm [15], ACO [16], Whale Optimization algorithm (WOA) [17], and Jellyfish Search Optimizer (JSO) [18] have been effectively modified to address issues with cloud work scheduling. PSO has been used for task scheduling in a cloud environment [19]. This research compares the work plans produced using techniques such as the Genetic Algorithm (GA), Brute Force (BF), First-In-First-Out (FIFO), and Delay Scheduling Policy (DSP). According to the experimental results, the PSO algorithm offers a more effective way to schedule tasks in a cloud environment than other approaches. It also gives notable benefits when it comes to optimizing the makespan value. An enhanced variant of Henry Gas Resolution Optimization, the Henry Gas-Harris Hawks-Comprehensive Contrast (HGHHC) approach, is suggested in [20]. The values of 34.30, 72.95 and 28.67 are for makespan and 16.92, 28.72 and 25.58 for resource utilization, respectively. Experimental results show that the HGHHC algorithm provides better simulated makespan and resource utilization compared to previous approaches. An adapted Chimpanzee Optimization Algorithm with improved exploration and exploitation stages is proposed to solve the task scheduling problem in cloud systems [21]. The success of the proposed method is tested for different simulation scenarios and the proposed method achieves a makespan improvement of approximately 30% compared to standard task scheduling algorithms. A method based on the Cloneable Jellyfish Algorithm is proposed for optimal task allocation in metaheuristic cloud environments [18]. The suggested algorithm's most innovative feature is its regulated dynamic population expansion, which helps to prevent local minima during the exploration stage. It also has a distinct cloning method to minimize the similarity between candidates in population growth. The experiments performed on the

Cloudsim simulator proved the success of the suggested approach in compared to traditional scheduling techniques.

Hybrid methods are used by combining different optimization techniques to provide more effective solutions addressing issues with cloud computing task scheduling [22]. These techniques, which are usually a combination of metaheuristic (e.g. GA, PSO, ACO, SA) and heuristic (e.g. RR, SJF) methods, provide higher efficiency and accuracy by utilizing the strengths of both methods. While some algorithms (e.g., SA and ACO) can get stuck in local minima, hybrid approaches minimize this problem by integrating multiple strategies [23]. Since cloud environments are dynamic, classical methods may not be able to adapt to changing workloads. However, hybrid algorithms can better adapt to this dynamism by combining metaheuristic and heuristic techniques [24]. A hybrid method is proposed using the Heterogeneous Earliest Finish Time(HEFT) algorithm [25]. According to simulation data, the suggested approach performs better in terms of makespan on random Direct Acyclic Graphs (DAG) than three heuristic and genetic algorithms. An approach to task scheduling that uses a hybrid optimization technique is presented [26]. The method considers other parameters such as minimum waiting time, overall production time, execution time, productivity and utilization in the scheduling of tasks. Results from simulations indicate that the suggested approach outperforms the traditional ACO and PSO-based scheduling algorithms in terms of performance. A hybrid algorithm is proposed by combining the ACO algorithm with the concept of gravitational search [27]. Simulation results performed with the CloudSim tool show that the proposed method outperforms ACO and the basic Gravitational Search Algorithm (GSA) [28].

The issue of task scheduling in cloud systems has garnered the interest of many researchers in the literature. A technique for building the PSO's initial population using heuristic methods is presented [29]. The authors use the Longest Job to Fastest Processor (LJFP) and Minimum Completion Time (MCT) to build the PSO population. The makespan, total execution time, degree of imbalance, and total energy consumption are used to assess the effectiveness of the suggested approaches. According on experimental findings, the suggested approaches outperform the conventional PSO. The drawbacks of PSO, including early convergence issues during the optimization process and becoming trapped in local minima, are addressed by using a Simulated Annealing approach [30]. According to experimental findings, the suggested approach can improve the ratio of average runtime to resource availability. A mathematical model known as the Load Balancing Mutation (LBMPSO), which considers availability and dependability, is proposed [31]. This model is a cloud computing particle swarm optimization (LBMPSO) scheduling technique that takes dependability, round-trip time, creation interval, execution time, transmission time, and load balancing between tasks and virtual machines into account. LBMPSO can improve the cloud computing environment's dependability by taking into account the resources that are available and delaying jobs that haven't been assigned yet. LBMPSO is contrasted with the randomized approach, standard PSO, and the Longest Cloud to Fastest Processor (LCFP) algorithm. Experimental results show that LBMPSO can lower execution times, round-trip times, and transmission costs.

Problem Definition and Methodology

Before being allocated to cloud virtual machines, user-submitted tasks are first put in the task queue. The task scheduler (TS) retrieves the specified number of tasks from the task queue and allocates them to the virtual machines according to the working principle. Tabular, matrix, graph, list, and coding techniques are commonly used in optimization methods. This study adopts the tabular method, illustrated in Fig.2, where tasks and virtual machines are represented. Here, the case of allocating 10 tasks to 3 virtual machines is represented. The tabular representation shown in Figure 2 represents a solution to be optimized. Fig. 3 shows a visualization of the execution of this representative solution on virtual machines. The makespan value is expressed as the highest of the amount of time needed for the tasks assigned to the servers to complete.

$VM/Tasks$	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
V_1	1	0	0	0	1	0	0	0	0	1
V_2	0	0	1	0	0	1	0	1	0	0
V_3	0	1	0	1	0	0	1	0	1	0

Figure 2. Solution Representation of A Task Scheduling.

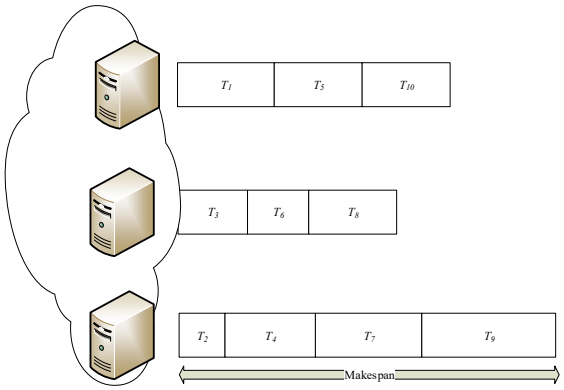


Figure 3. Makespan Representation.

Each task in the work queue is distinct and independent of the others. Each time a predetermined quantity of tasks are pulled from the queue of tasks and the tasks to be assigned are denoted by $T = \{T_1, T_2, .., T_i, .., T_n\}$. Here, t_k represents the quantity of instructions contained in the i'th task and n represents the number of tasks. The active virtual machines in the cloud to which the tasks will be assigned have different command processing capacities, and the total number of m active virtual machines in the system is denoted by $V = \{v_1, v_2, .., v_j, .., v_m\}$. v_l indicates how many million instructions per second (MIPS) that the l'th virtual machine processes. Consequently, the time of execution (ET_{lk}) of the k th task

on the l'st virtual machine is calculated by Equation (1). It is not necessary to allocate every task in the task queue to a separate machine. If whether the k'th task is allocated to the l'st virtual machine is expressed by a boolean parameter e_{lk}, then the entire amount of time needed to complete all tasks allocated to the j th virtual machine ($\mathcal{E}_l$) is found by Equation (2). When scheduling tasks in cloud systems, makespan is optimized by considering various objectives such as energy, cost, user service quality, task priority. In this study, optimization of the makespan value is performed. When using cloud systems, makespan (MS) expresses the greatest amount of time needed to complete a set of tasks across a collection of virtual machines and is represented by Equation (3).

$$ET_{lk} = \frac{t_k}{v_l} \quad (1)$$
$$\mathcal{E}_l = \sum_{k=0}^{N} e_{lk} * ET_{lk}$$

$$e_{lk} \begin{cases} 1, if\ k'th\ task\ \ was\ agined\ to\ l'th\ VM \\ 0,\ otherwise \end{cases} \quad (2)$$

$$MS = {}_{l=1}^{m}\, max(\mathcal{E}_l) \quad (3)$$

Particle Swarm Optimization

PSO is inspired by nature, in particular the collective movements of flocks of birds and groups of fish. The algorithm works by modelling the interactive movement of particles to scan a solution space. PSO uses a randomly initialized particle swarm to solve optimization problems. Each particle represents a point in the solution space and has two basic properties: position and velocity. Each particle moves by being influenced both by its own past experience (P_{best}) and by the best experience of other particles in the swarm (G_{best}) [1]. In this way, both exploration and exploitation processes are balanced. Equation (4-5) computes these particle movements. Here X_p and V_p are the particle's location and velocity values. $rand_1$ and $rand_2$ are the random coefficients. W is the particle's coefficient of inertia, which establishes the ratio of exploration to exploitation. c_1, c_2 are the particle motion's acceleration coefficients. In the representation form used for task scheduling, every vector index denotes a task, and the value contained within it corresponds to the VM's ID. The floating values' counterparts in integers were found by Equation (6) are considered since the VM ID is an integer. The scheduling vector is also these values (S_v). Algorithm 1 provides the PSO pseudo-code that was utilized in this investigation and modified for the task-scheduling issue. If any of the S_v's constituent parts value is greater than the upper and lower bound values, the appropriate component then selects an integer value between the lower and higher bounds that is created at random.

$$V_p = W * V_p + c_1 * rand_1 * \left(P_{best,i} - X_i\right) + c_2 * rand_2 * (P_{gbest} - X_i) \quad (4)$$
$$X_p{}^{new} = X_p + V_p \quad (5)$$
$$S_v = round\left(X_p{}^{new}\right) \quad (6)$$

Algorithm 1 The task-scheduling pseudo code of PSO

```
Initialize parameters
Initialize population
Itr_max ← maximum iteration number
N_pop ← Population size
For i=0 to N_pop
    F_i ←Calculate Fitness Value P_i
    If F_i > P_best
        P_best = F_i
G_best ← Particles with the highest fitness value overall
For itr=0 to Itr_max
  For i=0 to N_pop
    Determine the particle velocity using Equation 4.
    Particle position should be updated using Eq. 5.
    The position of particle is rounded to the closest value
    If (X_p < X_Lower ) or (X_p > X_Upper)
        X_p = Random(X_Lower, X_Upper)
```

Initialize population

Population generation in PSO is usually performed in a randomized manner. However, when the initially generated solutions are far from the optimum solution, this adversely affects the performance of the optimization process. The randomly generated solutions may be very similar to each other and may fall into undesired regions of the search space. This situation negatively affects the optimization process [22]. The heuristic method proposed in this study is realized by considering the population initially generated by PSO with various criteria. Firstly, the method calculates a load coefficient (LC) with Equation (7). This load factor is then multiplied by the capacities of the active virtual machines in the cloud and the amount of load that each server can take is calculated. This is as shown in Equation (8).

$$LC = \frac{\sum_k^n T_k}{\sum_l^m V_l} \quad 7$$

$$LD_k = V_k \times LC \quad \forall\ k\ \in \{1,2,3,\dots,m\} \quad 8$$

After determining the amount of load that can be assigned to all virtual machines, the tasks that will be subjected to scheduling are sorted from biggest to smallest based on how many instructions there are and allocated to virtual machines respectively. The reason for this is to reduce the makespan by prioritizing tasks with a high number of commands. When a task is assigned to a virtual machine, the current task load of the virtual machine must be greater than the time it takes to execute this task on this machine Equation (1). If the load of a virtual machine is greater than the time taken to execute a task, the task is allocated to this virtual machine and the new load of the virtual machine is

updated. If a task is not allocated to any virtual machine (if the current load of the virtual machines is less than the time it takes to execute the task), the task is allocated to the virtual machine with the minimum impact on the makespan time. Algorithm 2 displays the algorithm's pseudo code.

Algorithm 2 The pseudo-code of Heuristic approach for task-scheduling

```
Calculate Load Coefficient according to Eq7
LD← Calculate the workload that each virtual machine
can take according to Eq.8
T← Sort tasks from largest to smallest by command
V← CPU capacities of all VM's
For i to Size(T)
    IsAssign=False
    For j to Size(V)
        ET← Calculate according to Eq.1
        If (LD(i)>ET)
            Assign task i to VM j
            LDi= LDi -ET
            IsAssign=True
    If (!IsAssign)
        Assign task to Vm according to minimum
                    makespan
```

Results and Discussion

The success of the proposed method is tested in the CloudSim environment considering different scenarios [32]. In the simulations, the performance of Heuristic PSO is presented comparatively with FCFS, ACO, and PSO in terms of makespan and energy consumption. To evaluate the performance of proposed method used scenarios with varying quantities of tasks and virtual machines (VMs). The computational capacity of each VM is randomly assigned to reflect the heterogeneity of real cloud environments. Similarly, the quantity of task instructions was also randomized to simulate loads of different complexity.

Two main performance metrics were considered:

Makespan

The overall task completion times were measured to assess scheduling efficiency. The results indicate that Heuristic-PSO consistently achieved a shorter makespan compared to FCFS, ACO, and standard PSO. This demonstrates its ability to allocate tasks more effectively across available resources, leading to balanced load distribution and improved scheduling efficiency.

Energy Consumption

Energy usage was analyzed by calculating the power consumed by VMs during both idle and active states. The findings show that Heuristic-PSO reduced total energy consumption more effectively than the benchmark algorithms. This improvement highlights the capability of the proposed method to minimize energy expenditure while maintaining scheduling performance.

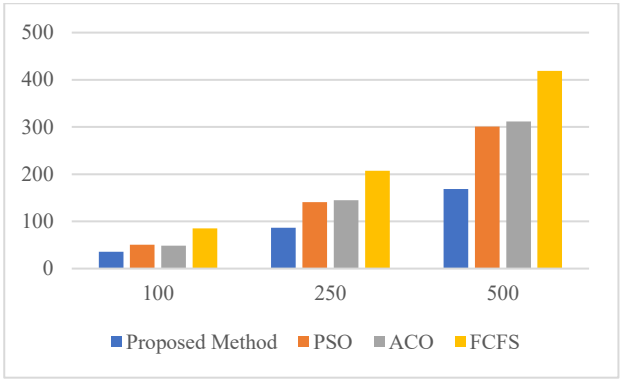
Overall, the experimental results confirm that integrating heuristic strategies into PSO not only enhances task scheduling efficiency but also improves energy awareness in cloud environments.The number of instructions and virtual machine processor capacities of the tasks used in each scenario were performed randomly. The amount of energy spent for instruction execution was assigned in proportion to the processor capacity of each virtual machine. In addition, the amount of energy consumed by the virtual machines when idle was taken as 30% of the amount of energy at load. Thus, the amount of energy consumed by the servers when they are idle is also taken into account in a task scheduling. In this study, the results obtained for each scenario are averaged over 100 independent simulations. The simulations were run on a machine with an Intel(R) Core(TM) i5-10400 CPU running at 2.90 GHz and 8.00 GB of RAM. Table 1 lists other simulation parameters.

Table 1 The simulations parameters

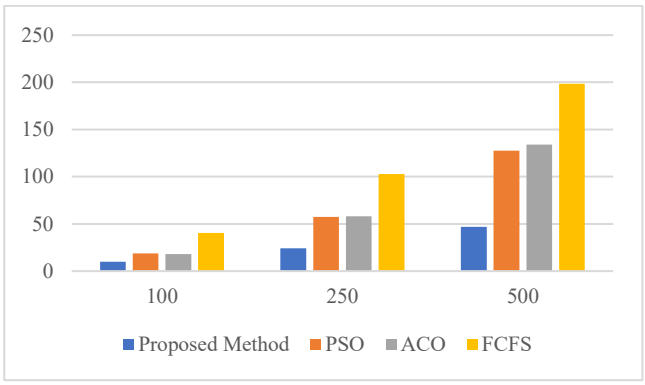
CloudSim	Number of tasks	100-250-500
	Number of task commands	4.000-6.000
	VM numbers	5-10-20
	MIPS	1 000-20 000
	Maximum Iteration	100
	Number of Particles and Ant	20
PSO	c_1	1
	c_2	2
	w	0.5
	Maximum Iteration	100
ACO	Ant Numbers	20
	α	1
	β	2
	σ	0.1

In the simulations, the first performance evaluation was carried out in terms of makespan. The average makespan findings from 100 separate simulations are displayed in Fig. 4, and the statistical summaries of these experimental data are displayed in Table 2. The results shown in Fig. 4 and Table 2 indicate that Heuristic PSO produces lower makespan values compared to FCFS, ACO and PSO algorithms. Especially in scenarios where the number of virtual machines and the amount of tasks increases, Heuristic PSO provides more stable and lower makespan times. In the five virtual machine scenarios, Heuristic PSO produced shorter makespan values than FCFS, ACO and PSO for all task numbers. The most significant difference was observed in the scenario with 500 tasks. In scenarios with 5 virtual machines, the average makespan value of Heuristic PSO was 168.63, while that of standard PSO was 301, ACO was 312.03 and FCFS was 419.07. In scenarios with 10 virtual machines, Heuristic PSO achieved very good results in terms of makespan. In scenarios with 20 virtual machines, the average makespan value of Heuristic

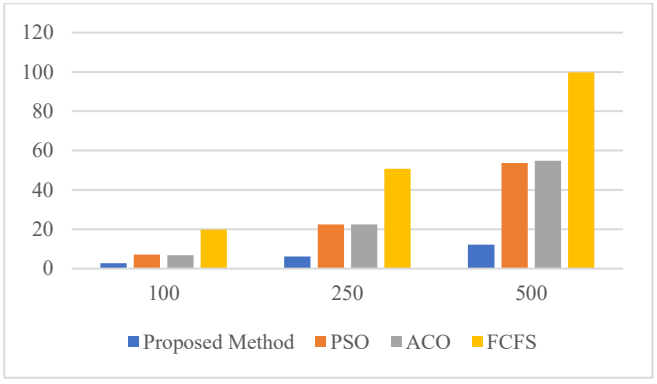
PSO was determined as 47.01, while it was calculated as 127.66 for standard PSO. In the 20 virtual machine scenarios, increasing the number of virtual machines generally decreased the makespan times, but Heuristic PSO again achieved better results compared to FCFS, ACO and PSO. In the 500 task scenario, the average makespan value of Heuristic PSO was 12.26, while standard PSO was 53.63. These results show that Heuristic PSO can significantly reduce the makespan time by distributing the tasks more evenly. Especially in large-scale systems, the proposed method minimizes the early convergence problem and achieves a better scheduling success.



A. Average makespan results for scenarios with 5 VMs



B. Average makespan results for scenarios with 10 VMs



C. Average makespan results for scenarios with 20 VMs

Figure 4. Average makespan outcomes across all techniques

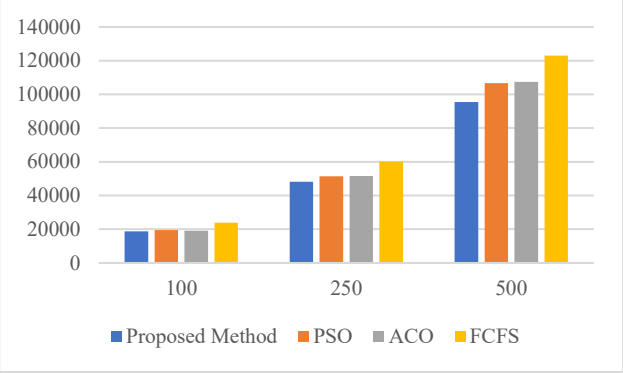
Table 2. Statistical results of makespan simulations

Scenarios		Heuristic PSO				PSO				ACO				FCFS			
VM	**Task**	**Mean**	**Min**	**Max**	**Median**	**Mean**	**Min**	**Max**	**Median**	**Mean**	**Min**	**Max**	**Median**	**Mean**	**Min**	**Max**	**Median**
5	**100**	35.80	23.77	84.25	33.56	50.71	25.41	93.40	49.93	48.79	24.01	88.52	46.25	85.06	32.11	105.89	99.97
	250	86.49	52.32	182.87	83.70	140.59	53.69	252.85	143.65	144.85	53.55	227.93	149.52	207.12	61.99	261.64	247.98
	500	168.63	108.91	250.32	165.90	301.00	122.40	496.25	301.59	312.03	116.57	473.23	350.04	419.07	128.04	513.84	497.82
10	**100**	9.80	7.30	17.53	9.38	18.62	8.62	38.70	17.10	18.09	8.59	36.40	16.64	40.48	15.35	54.28	49.71
	250	24.05	16.60	42.62	23.39	57.33	21.86	99.41	59.09	58.22	19.93	105.62	57.78	102.79	24.77	132.65	123.52
	500	47.01	34.51	97.53	44.10	127.66	48.77	227.60	128.73	134.09	48.59	226.55	140.29	198.49	64.97	259.04	248.42
20	**100**	2.74	2.03	4.07	2.67	7.13	2.96	16.58	6.59	6.89	3.23	15.70	6.41	19.85	5.44	28.30	24.02
	250	6.18	4.62	8.14	6.09	22.52	8.14	51.76	19.93	22.46	7.79	51.13	19.92	50.72	13.53	68.15	60.34
	500	12.26	8.77	16.52	12.32	53.63	14.1	108.76	49.45	54.83	14.21	105.22	49.34	99.73	24.98	133.61	124.00

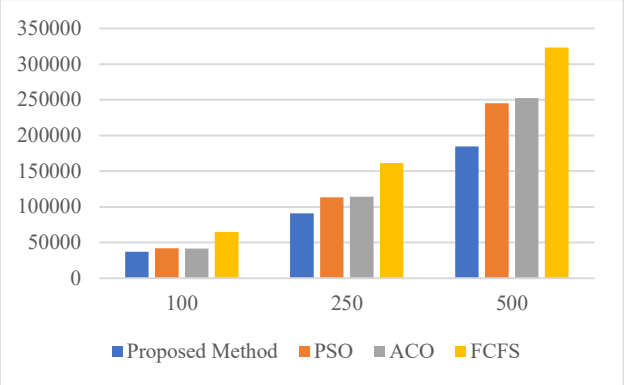
Table 3. Statistical results of energy consumption (joules) simulations

Scenarios		Heuristic PSO				PSO				ACO				FCFS			
VM	**Task**	**Mean**	**Min**	**Max**	**Median**	**Mean**	**Min**	**Max**	**Median**	**Mean**	**Min**	**Max**	**Median**	**Mean**	**Min**	**Max**	**Median**
5	**100**	18 690.3	6 919.7	2 4654.0	19 333.1	19 495.4	6 784.5	28 578.1	19 625.5	19 163.7	6 771.3	27 700.8	19 674.1	23 871.1	6 618.8	42 295.4	22 434.4
	250	48 115.7	21 081.3	63 845.6	49 025.6	51 481.5	20 850.6	76 778.8	52 200.0	51 678.7	21 001.8	78 312.0	51 636.8	60 201.6	21 018.8	104 409.7	59 685.3
	500	95 475.5	62 567.4	124 900.5	95 743.1	106 651.4	65 794.3	171 799.0	105 962.7	107 530.3	65 453.6	162 155.4	105 187.1	122 945.9	70 608.8	194 945.4	117 098.1
10	**100**	36 847.2	22 706.3	48 263.5	37 372.1	42 167.0	23 520.3	68 351.5	40 709.3	41 517.2	23 269.0	60 734.8	40 012.9	64 761.8	35 269.4	110 941.8	60 742.9
	250	90 759.4	56 760.2	117 986.3	90 927.7	113 509.2	58 428.5	187 464.5	106 698.0	114 175.9	57 483.5	182 402.6	107 246.9	161 196.2	74 495.4	318 193.7	152 883.4
	500	184 656.9	99 382.6	220 893.9	186 892.5	245 162.9	113 312.2	404 191.6	236 176.3	252 298.6	110 292.4	438 957.3	240 085.4	323 368.8	120 371.3	581 132.8	304 918.5
20	**100**	74 632.7	52 081.0	114 731.7	74 655.6	100 427.9	62 762.8	177 449.2	94 736.1	98 650.4	62 458.4	167 559.9	94 229.7	205 792.7	76 528.7	368 780.9	207 398.9
	250	182 657.2	147 394.1	219 206.7	184 468.2	291 055.9	155 899.2	52 6540.8	279 832.3	289 231.4	160 443.7	508 569.9	285 331.8	531 021.1	197 144.0	845 138.7	517 731.8
	500	363 855.1	286 992.2	430 766.6	363 908.9	643 860.2	351 918.9	126 7283.0	615 555.9	650 094.9	364 609.0	1 285 109.0	612 781.6	1 028 482	463 413.9	1 677 298.0	1 065 899.0

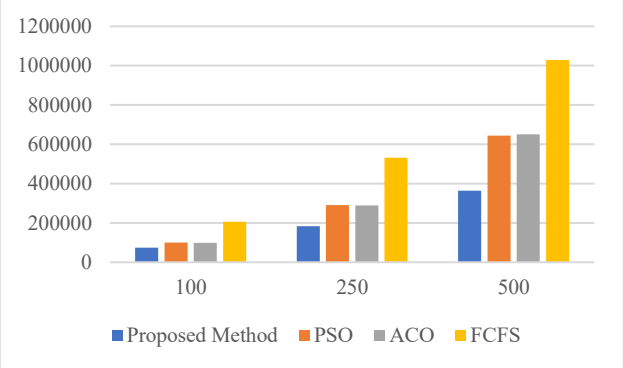
The second performance evaluation was conducted by looking at how much energy the cloud environment's virtual machines used. The average energy consumption findings from the separate tests are displayed in Fig. 5, and the statistical summaries of these experimental results are displayed in Table 3. Fig. 5 and Table 3 show that Heuristic PSO is more efficient in terms of energy consumption compared to FCFS, ACO and PSO. In the 5 virtual machine scenarios, Heuristic PSO achieved lower energy consumption at all task numbers. For example, in the scenario with 500 tasks, the average energy consumption of Heuristic PSO was calculated as 95475.49 units, while the consumption of standard PSO was 106651.44 units. In 10 virtual machine scenarios, energy consumption decreased with increasing the number of virtual machines. The energy consumption of Heuristic PSO in the scenario with 500 tasks was 184656.92 units, while the standard PSO, which is the closest algorithm, showed a consumption of 245162.91 units. In the 20 virtual machine scenarios, the presence of more virtual machines further increased the efficiency of Heuristic PSO. In the case with 500 tasks, the energy consumption of Heuristic PSO was calculated as 363855.14 units, while the consumption of standard PSO reached 643860.2 units. These results show that Heuristic PSO optimizes not only the makespan time but also the energy consumption by distributing the tasks more evenly. Especially in large-scale systems, the proposed method achieves a more efficient scheduling success by minimizing the problem of excessive energy consumption compared to other algorithms.



A. Average energy consumption results for scenarios with 5 VMs



B. Average energy consumption results for scenarios with 10 VMs



C. Average energy consumption results for scenarios with 20 VMs

Figure 5. Average energy consumption outcomes across all techniques

Conclusion

In this study, a heuristic PSO-based approach is proposed to solve the task scheduling problem in cloud environments. The proposed method provides a more balanced task distribution by improving the initial population of PSO with a heuristic mechanism. The extensive simulations show that the heuristic PSO reduces the makespan time by 61.42% and 62.84% on average compared to the standard PSO and ACO algorithms. It was also found to consume 26.18% less energy than PSO and 27.33% less energy than ACO. Especially in large systems, heuristic PSO provides a more balanced distribution of tasks by minimizing the early convergence problem. The results show that the proposed method offers a more efficient, scalable and energy-saving alternative for task scheduling in cloud computing. In the future, it is planned to further develop the method with multi-objective optimization approaches, to integrate it with dynamic load balancing strategies and to test it in large-scale applications such as big data processing.

Ethics committee approval and conflict of interest statement

There is no need to obtain permission from the ethics committee for the article prepared

There is no conflict of interest with any person / institution in the article prepared

Authors' Contributions

Barut Cebrail: Study conception and design, visualization, analysis, and interpretation of data, drafting of manuscript

Firildak Kazım: conceived the original idea, supervised the project, critical revision

References

[1] C. Barut, G. Yildirim, and Y. Tatar, "An intelligent and interpretable rule-based metaheuristic approach to task scheduling in cloud systems," *Knowledge-Based Syst.*, vol. 284, no. December 2023, p. 111241, 2024, doi: 10.1016/j.knosys.2023.111241.

[2] G. Boss, P. Malladini, D. Quan, L. Legregni, and H. Hall, "Cloud Computing Authors:," *Cloud Comput.*, vol. 17, no. 1, pp. 111–136, 2012.

[3] Y. Pachipala, K. S. Sureddy, A. B. S. Sriya Kaitepalli, N. Pagadala, S. S. Nalabothu, and M. Iniganti, "Optimizing Task Scheduling in Cloud Computing: An Enhanced Shortest Job First Algorithm," *Procedia Comput. Sci.*, vol. 233, no. 2023, pp. 604–613, 2024, doi: 10.1016/j.procs.2024.03.250.

[4] A. Keivani and J. R. Tapamo, "Task scheduling in cloud computing: A review," *icABCD 2019 - 2nd Int. Conf. Adv. Big Data, Comput. Data Commun. Syst.*, pp. 1–6, 2019, doi: 10.1109/ICABCD.2019.8851045.

[5] A. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Futur. Gener. Comput. Syst.*, vol. 91, pp. 407–415, 2019, doi: 10.1016/j.future.2018.09.014.

[6] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Ind. Electron. Handb. - Five Vol. Set*, pp. 1942–1948, 2011, doi: 10.1007/978-3-319-46173-1_2.

[7] A. S. Abohamama, A. El-Ghamry, and E. Hamouda, *Real-Time Task Scheduling Algorithm for IoT-Based Applications in the Cloud–Fog Environment*, vol. 30, no. 4. Springer US, 2022. doi: 10.1007/s10922-022-09664-6.

[8] P. Y. Zhang and M. C. Zhou, "Dynamic Cloud Task Scheduling Based on a Two-Stage Strategy," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 772–783, 2018, doi: 10.1109/TASE.2017.2693688.

[9] C. Barut, G. Yildirim, and Y. Tatar, "An intelligent and interpretable rule-based metaheuristic approach to task scheduling in cloud systems," *KNOWLEDGE-BASED Syst.*, vol. 284, 2024, doi: 10.1016/j.knosys.2023.111241.

[10] H. Jin *et al.*, "A survey of energy efficient methods for UAV communication," *Veh. Commun.*, vol. 41, p. 100594, 2023, doi: 10.1016/j.vehcom.2023.100594.

[11] P. Banerjee, A. Tiwari, B. Kumar, K. Thakur, A. Singh, and M. Kumar Dehury, "Task Scheduling in cloud using Heuristic Technique," *7th Int. Conf. Trends Electron. Informatics, ICOEI 2023 - Proc.*, no. Icoei, pp. 709–716, 2023, doi: 10.1109/ICOEI56765.2023.10126030.

[12] K. Beghdad Bey, F. Benhammadi, and R. Benaissa, "Balancing heuristic for independent task scheduling in cloud computing," *12th Int. Symp. Program. Syst. ISPS 2015*, pp. 7–12, 2015, doi: 10.1109/ISPS.2015.7244959.

[13] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing," *Sensors*, vol. 22, no. 3, pp. 1–22, 2022, doi: 10.3390/s22030920.

[14] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, and M. Zivkovic, "Task Scheduling in Cloud Computing Environment by Grey Wolf Optimizer," *27th Telecommun. Forum, TELFOR 2019*, no. June 2020, 2019, doi: 10.1109/TELFOR48224.2019.8971223.

[15] S. Mangalampalli, G. R. Karri, and A. A. Elngar, "An Efficient Trust-Aware Task Scheduling Algorithm in Cloud Computing Using Firefly Optimization," *Sensors*, vol. 23, no. 3, 2023, doi: 10.3390/s23031384.

[16] S. Asghari and N. J. Navimipour, "Cloud service composition using an inverted ant colony optimisation algorithm," *Int. J. Bio-Inspired Comput.*, vol. 13, no. 4, p. 257, 2019, doi: 10.1504/IJBIC.2019.100139.

[17] X. Chen *et al.*, "A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3117–3128, 2020, doi: 10.1109/JSYST.2019.2960088.

[18] M. Bürkük and G. Yıldırım, "Cloneable Jellyfish Search Optimizer Based Task Scheduling in Cloud Environments," *Türk Doğa ve Fen Derg.*, vol. 11, no. 3, pp. 35–43, 2022, doi: 10.46810/tdfd.1123962.

[19] S. H. Adil, K. Raza, U. Ahmed, S. S. A. Ali, and

M. Hashmani, “Cloud task scheduling using nature inspired meta-heuristic algorithm,” *ICOSST 2015 - 2015 Int. Conf. Open Source Syst. Technol. Proc.*, pp. 158–164, 2016, doi: 10.1109/ICOSST.2015.7396420.

[20] N. O. Alkaam, A. B. M. Sultan, M. B. Hussin, and K. Y. Sharif, “Hybrid Henry Gas-Harris Hawks Comprehensive-Opposition Algorithm for Task Scheduling in Cloud Computing,” *IEEE Access*, vol. 13, no. January, pp. 12956–12965, 2025, doi: 10.1109/ACCESS.2025.3530860.

[21] E. GÜNDÜZALP, G. YILDIRIM, and Y. TATAR, “Efficient Task Scheduling in Cloud Systems with Adaptive Discrete Chimp Algorithm,” *Balk. J. Electr. Comput. Eng.*, vol. 10, no. 3, pp. 328–336, 2022, doi: 10.17694/bajece.989467.

[22] B. F. Azevedo, A. M. A. C. Rocha, and A. I. Pereira, *Hybrid approaches to optimization and machine learning methods: a systematic literature review*, vol. 113, no. 7. Springer US, 2024. doi: 10.1007/s10994-023-06467-x.

[23] N. Mansouri, B. Mohammad Hasani Zade, and M. M. Javidi, “Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory,” *Comput. Ind. Eng.*, vol. 130, no. July 2018, pp. 597–633, 2019, doi: 10.1016/j.cie.2019.03.006.

[24] Sandeep Kumar Patel and Avtar Singh, “Task scheduling in cloud computing using hybrid optimization algorithm,” *Soft Comput.*, vol. 26, no. 23, pp. 13069–13079, 2022, doi: 10.1007/s00500-021-06488-5.

[25] A. Kamalinia and A. Ghaffari, “Hybrid Task Scheduling Method for Cloud Computing by Genetic and DE Algorithms,” *Wirel. Pers. Commun.*, vol. 97, no. 4, pp. 6301–6323, 2017, doi: 10.1007/s11277-017-4839-2.

[26] M. S. A. Khan and R. Santhosh, “Task scheduling in cloud computing using hybrid optimization algorithm,” *Soft Comput.*, vol. 26, no. 23, pp. 13069–13079, 2022, doi: 10.1007/s00500-021-06488-5.

[27] S. Rani and P. K. Suri, “An efficient and scalable hybrid task scheduling approach for cloud environment,” *Int. J. Inf. Technol.*, vol. 12, no. 4, pp. 1451–1457, 2020, doi: 10.1007/s41870-018-0175-3.

[28] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, “GSA: A Gravitational Search Algorithm,” *Inf. Sci. (Ny).*, vol. 179, no. 13, pp. 2232–2248, 2009, doi: 10.1016/j.ins.2009.03.004.

[29] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, “Heuristic initialization of PSO task scheduling algorithm in cloud computing,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2370–2382, 2022, doi: 10.1016/j.jksuci.2020.11.002.

[30] S. Zhan and H. Huo, “Improved PSO-based task scheduling algorithm in cloud computing,” *J. Inf. Comput. Sci.*, vol. 9, no. 13, pp. 3821–3829, 2012.

[31] A. I. Awad, N. A. El-Hefnawy, and H. M. Abdel-Kader, “Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments,” *Procedia Comput. Sci.*, vol. 65, no. Iccmit, pp. 920–929, 2015, doi: 10.1016/j.procs.2015.09.064.

[32] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and Rajkumar Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms Rodrigo,” *Softw. - Pract. Exp.*, vol. 39, no. 7, pp. 701–736, 2009, doi: 10.1002/spe.