SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

HEFT-Guided Elite PSO for Workload Scheduling in Cloud Computing

Abhishek Bishnoi[1], Jai Bhagwan[1]*, Sanjeev Kumar[1]

[1]Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar – 125001, India
*Corresponding Author E-mail: drjaicse@gmail.com, ORCID: 0000-0002-8708-4029

Abstract

Efficient workflow scheduling remains a critical challenge in cloud computing due to precedence constraints among dependent tasks and heterogeneous nature of virtual machines. The traditional heuristic scheduling approaches often fail to produce optimum solutions as the workflows scheduling an NP-hard optimization problem. The conventional swarm intelligence algorithms suffer from premature convergence and slow searching problems. So, this paper proposes a HEFT-Guided Elite Particle Swarm Optimization (EPSO) algorithm to improve workflows scheduling problem in heterogeneous cloud environment. The proposed algorithm combines the HEFT policy for VMs initialization, elite class solution preservation to move towards better scheduling solutions and adaptive strategies for self-decision-making during evolution process for exploration and exploitation. The proposed EPSO algorithm is evaluated using four standard workflows datasets namely CyberShake, Montage, Inspiral and Epigenomics. The performance of the proposed EPSO algorithm has been compared with APSO, IPSO, PSOJaya, PSO and GWO algorithm. The EPSO algorithm found better in terms of makespan, cost, degree of imbalance i.e. load distribution among VMs and through put.

Keyword: Cloud Computing, Cost, Elite Preservation, Makespan, Particle Swarm Optimization, EPSO, APSO, Virtual Machines (VMs)

1 Introduction

The rapid expansion of cloud computing has fundamentally changed the execution of computationally intensive applications by providing scalable, on-demand computing resources through virtualization. Scientific communities in astronomy, bioinformatics, climate modelling, earthquake engineering, genomics, and high-energy physics increasingly rely on cloud infrastructures to execute complex workflow applications that consist of hundreds or even thousands of interdependent tasks [1][2]. Unlike independent task scheduling, workflow scheduling requires preserving task dependencies while efficiently utilizing heterogeneous computing resources. As cloud providers continue to support increasingly data-intensive applications, the design of efficient workflow scheduling algorithms has become a critical research challenge. Directed Acyclic Graphs (DAGs), in which each node represents a computational job and each directed edge indicates a data dependence between two tasks, are frequently used to model scientific processes. Workflow scheduling is far more difficult than scheduling individual tasks since a job cannot start execution until all of its predecessors have finished satisfactorily [3]. In addition to allocating tasks to available virtual machines (VMs) and determining the order in which they should be executed, the scheduler must also take into account limitations on workflow precedence, communication delays, computation costs, and resource heterogeneity. For actual workflow sizes, achieving an ideal plan by exhaustive search becomes computationally prohibitive because workflow scheduling falls within the family of NP-hard optimization problems [4].

Many heuristic policies have been invented to solve workflow scheduling issue in heterogeneous computing systems within last two decades. Because of easy to use, maintaining parent child dependency, computational effectiveness, the HEFT (Heterogeneous Earliest Finish Time) algorithm is popular to solve scheduling issues [5]. It uses and upward ranking approach to set priorities of tasks available in a workflow. After setting priorities it allows to map these tasks to virtual machines effectively. HEFT algorithm is a deterministic greedy approach in which scheduling decisions are made sequentially and are occasionally reconsidered once allocated while producing high-quality schedules. The scheduler may be prevented from exploring alternate task-resource

mappings that might result in superior global solutions when workflow size and resource heterogeneity rise due to HEFT's greedy nature [6]. As a result, the academics have looked for optimization strategies that can go beyond deterministic scheduling choices.

Population-based metaheuristic algorithms do global search across vast solution spaces without requiring gradient information. These have become strong alternatives to heuristic algorithms. Particle Swarm Optimization (PSO) is one of these algorithms that has attracted a lot of interest because of its straightforward mathematical model and easy implementation [7]. PSO iteratively enhances candidate solutions by exchanging information between individual particles and the swarm's top performers. It works by taking inspiration from the cooperative movement of fish schools and bird flocks. Because of these features the PSO is especially well-suited for resolving complex scheduling issues with several competing goals. Conventional PSO has a number of well-known drawbacks when it comes to workflow scheduling despite its widespread use. Particles have a tendency to assemble in locally favourable areas in the later stages of optimization which decreases swarm diversity and raises the risk of premature convergence [8], although it is a faster algorithm. In high-dimensional scheduling situations when there are many possible task-resource pairings, this behaviour is particularly noticeable. Additionally, poor-quality starting populations are frequently produced by random initialization. It necessitates more iteration before the swarm makes approaches competitive scheduling solutions. Consequently for complex scientific operations, the ultimate scheduling quality and convergence speed may get declined. Recent research has looked into hybrid and adaptive optimization strategies that combine the global search capabilities of swarm intelligence techniques. These strategies have the advantages of heuristic scheduling algorithms in order to address these drawbacks [9]–[12]. In these methods, the optimization process is directed toward promising areas of the search space by deterministic heuristics which produce an initial viable schedule. By reducing needless exploration during early iterations this informed initialization frees up the optimization process to concentrate on fine-tuning high-quality schedules. The Makespan, execution cost, and resource usage have all improved using a number of hybrid techniques based on HEFT, GAs, Differential Evolution, Ant Colony Optimization, and Grey Wolf Optimization [13]–[16]. However, a lot of current hybrid approaches either rely on predefined evolutionary parameters or are unable to efficiently maintain high-quality solutions during the optimization process. Adding elite learning mechanisms to swarm optimization is another intriguing approach. Elite techniques allow the search process to take advantage of high-quality areas while ensuring enough exploration of the remaining population by preserving superior candidate solutions throughout subsequent generations [17]. These techniques have shown promising performance in a number of optimization fields due to speeding up convergence and enhancing solution stability. However, their use in scientific workflow scheduling is still very restricted especially when paired with heuristic starting techniques.

Taking inspiration from these findings, the HEFT-Guided Elite Particle Swarm Optimization (EPSO) technique for workflow scheduling in heterogeneous cloud computing settings is proposed in this research. The proposed method uses the HEFT algorithm to create starting schedules that meet workflow precedence criteria rather than creating a random swarm. The particles can start the search from potential places rather than random locations. These schedules act as direction for the optimization process. Additionally, an elite preservation technique is implemented to maintain better scheduling solutions in succeeding generations. It enhances convergence behaviour and lowers the possibility of early stagnation of the proposed algorithm. The introduced technique seeks to improve workflow execution performance in terms of makespan, execution cost, and effective resources usages by fusing heuristic scheduling with elite-guided swarm optimization.

The primary contributions of this research work are summarized as follows.

1. A hybrid scheduling algorithm that fuses HEFT initialization with Elite Particle Swarm Optimization has been designed.
2. An elite preservation policy along with adaptive PSO improves population quality and enhances convergence during optimization process.
3. An efficient workflow scheduling strategy that maintains task precedence while exploring more task–VM mappings.
4. Comparison of proposed EPSO with other algorithms using standard scientific workflow datasets, including CyberShake, Montage, Inspiral, and Epigenomics to check effectiveness.

The rest of this paper is organized as follows. Section 2 reviews recent developments in workflow scheduling and hybrid optimization techniques. Section 3 formulates the workflow scheduling problem and presents the system model. Section 4 describes the proposed HEFT-Guided Elite PSO algorithm in detail. Section 5 discusses the

experimental setup and performance evaluation, while Section 6 concludes the paper and outlines future research directions.

2 Literature Review

Because workflow scheduling directly affects execution efficiency, resource usage, service cost, and overall Quality of Service (QoS), it continues to be one of the most active study areas in heterogeneous and cloud computing. Workflow scheduling necessitates preserving precedence connections among dependent activities while concurrently utilizing diverse computer resources, in contrast to independent task scheduling. As a result, several heuristic and metaheuristic algorithms have been put out to enhance the performance of process execution. Deterministic heuristic approaches were the main focus of early workflow scheduling research due to their predictable scheduling behaviour and minimal computer complexity. One of the most effective list scheduling methods for heterogeneous systems is the Heterogeneous Earliest Finish Time (HEFT) algorithm, which was put out by Topcuoglu et al. [5]. HEFT assigns each job to the processor that minimizes its earliest finish time after determining task priority through upward ranking. It is the standard approach for workflow scheduling research due to its steady performance and computational economy. Further research tried to improve the HEFT by implementing look-ahead scheduling solution principles to improve resource utilization. The task duplication methods and optimistic cost tables were used [12]-[14]. These expansions maintained the greedy scheduling philosophy making them vulnerable to less-than-ideal global decisions even when they improved scheduling quality under certain circumstances. Researchers started investigating evolutionary and swarm intelligence systems that could search far wider solution areas as workflow applications grew more complicated. Workflow scheduling issues have been successfully solved using a number of nature-inspired or swarm intelligence techniques including Genetic Algorithms (GAs), Differential Evolution (DE), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Grey Wolf Optimization (GWO), Whale Optimization Algorithm (WOA), and several others [15]–[18]. These algorithms iteratively refine candidate schedules rather than depending on a single greedy assignment. They often offer superior global search capabilities than deterministic heuristics. Particle Swarm Optimization has drawn a special interest among swarm intelligence methods because of its straightforward mathematical formulation and minimal parameter constraints. Pandey et al. [6] presented that by taking computation and communication delays into account together, The PSO may efficiently reduce process execution costs. Subsequent research improved convergence behaviour and decreased the likelihood of early stagnation by using nonlinear inertia weights. The adaptive acceleration coefficients, constriction factors, and dynamic neighborhood structures were also considered [19]–[22]. After comparing to the original PSO model, these changes greatly improved optimization performance especially for medium-sized workflow applications. Despite these advancements the traditional PSO-based workflow scheduling still has a number of limitations. The first swarm is created at random without taking resource availability or process features into account. As a result, before particles reach suitable scheduling areas a significant number of iterations are frequently needed. Additionally, the population eventually becomes less diverse when optimization moves forward. The makes leading particles to congregate around locally optimum solutions. When the search space grows exponentially in scientific processes with hundreds or thousands of dependent activities, this issue becomes more relevant [20]- [23]. Because of its distributed search capacity and positive feedback mechanism, Ant Colony Optimization has also been thoroughly studied for workflow scheduling. Artificial ants can build scheduling solutions gradually while taking use of previously identified high-quality assignments thanks to pheromone trails. When compared to traditional scheduling algorithms, some ACO variations have shown advantages in workflow makespan, execution cost, and load balancing [23]–[25]. However, excessive pheromone accumulation may eventually restrict search variety, and ACO often takes a reasonably high number of cycles before pheromone information becomes suitably useful. Another comparable option for cloud process scheduling is Grey Wolf Optimization. An efficient balance between exploration and exploitation during optimization is made possible by GWO's hierarchical leadership system. GWO often performs better than traditional PSO in avoiding local optima while generating competitive scheduling quality, according to several comparative studies [26][27]. However, ordinary GWO's local exploitation capacity is rather poor close to convergence, which encourages researchers to combine GWO with complementing optimization methods.

As a result, recent studies have moved toward hybrid optimization frameworks that blend population-based optimization algorithms with deterministic heuristics. Heuristic algorithms are initially used to create workable schedules that meet workflow precedence limitations rather than creating entirely random beginning populations. The evolutionary optimization process is then directed toward potential areas of the search space by these heuristic solutions. When compared to standalone swarm-based techniques the hybrid systems combining HEFT with Genetic Algorithms, Differential Evolution, PSO, and ACO have consistently shown faster convergence and better scheduling performance [28]–[31]. The high quality starting of the schedulers minimize

needless exploration during early optimization while maintaining adequate flexibility for later upgrades. The integration of adaptive learning techniques into swarm optimization algorithms is another significant advancement in today's technical era of IoT. To dynamically control exploration and exploitation throughout the optimization process, researchers have suggested adaptive inertia weights, self-adjusting learning coefficients, chaotic initialization, opposition-based learning, Levy flight operators and local search methods [19][21][32]. These adaptive methods lessen sensitivity to parameter selection and increase convergence stability during optimization process. However, the majority of current methods keep updating all particles in the same way regardless of the quality of the search which may lead to the elimination of extremely promising scheduling solutions in subsequent generations.

By maintaining high-quality candidate solutions while enabling the remaining population to keep investigating different scheduling areas, the elite learning algorithms offer a practical way to overcome this constraint. Because elite preservation speeds up convergence without appreciably compromising population diversity as a result it has shown promising results in a number of optimization contexts [33][34]. Despite these benefits, elite-guided swarm evolution and heuristic initialization are rarely explicitly combined in workflow scheduling systems. The majority of hybrid algorithms now in use either use inflexible elite selection procedures that are unable to properly utilize the information included in high-quality schedules or only maintain a single best solution. Additionally, a review of recent research shows that workflow scheduling studies still primarily concentrate on minimizing makespan or execution cost separately whereas the real-world cloud environments necessitate the simultaneous optimization of several QoS objectives such as execution time, resource utilization, energy consumption, load balancing along with scheduling stability. The necessity for scheduling algorithms that may achieve fast convergence without sacrificing global search capabilities is further highlighted by the growing complexity of scientific operations and the increasing variety of cloud infrastructures.

2.1 Research Gap

Despite the fact that several workflow scheduling methods have been put out, a number of significant issues are still unsolved. First, after resource assignments have been set, deterministic heuristics like HEFT provide effective initial schedules but have little capacity to improve them. Second, when scheduling big scientific processes, traditional PSO and many of its variations still rely on randomly started populations, which cause slower convergence and a higher likelihood of premature standstill. Third, incorporating HEFT-generated elite schedules directly into the swarm evolution process has received very little attention, despite the fact that hybrid scheduling techniques have shown enhanced performance. Current research often only uses heuristic initialization once and does not take advantage of elite scheduling expertise during the optimization process. Lastly, a lot of algorithms use makespan and cost to assess scheduling performance, with very little focus on maintaining high-quality scheduling patterns across generations. The HEFT-Guided Elite Particle Swarm Optimization (HEFT-EPSO) technique, which combines elite-guided evolutionary learning with informed heuristic initialization to enhance workflow scheduling efficiency in diverse cloud settings, was developed in response to these discoveries.

3 Problem's Formulation

Scientific workflow scheduling aims to allocate a set of dependent tasks onto heterogeneous virtual machines while satisfying precedence constraints and optimizing multiple Quality of Service (QoS) objectives. In cloud computing environments, workflows are generally represented as a Directed Acyclic Graph (DAG), denoted by

$$w = (T, E) \tag{1}$$

where $T = \{t_1, t_2, \dots, t_n\}$ represents the set of workflow tasks and E denotes the dependency edges among tasks. An edge $\{t_i, t_j\}$ indicates that task T_j cannot begin execution until task t_j has completed. Therefore, every scheduling decision must preserve workflow precedence constraints throughout the execution. Let:

$$VM = \{vm_1, vm_2, \dots, vm_m\} \tag{2}$$

be the available heterogeneous virtual machines as discussed in simulation setup. The objective of the scheduler is to determine an efficient mapping

$$f: T \rightarrow VM \tag{3}$$

such that every workflow task is assigned to exactly one virtual machine while minimizing workflow completion time and execution cost, degree of imbalance.

3.1 Makespan

The overall workflow completion time, commonly referred to as makespan, is defined as the finishing time of the last completed workflow task i.e. maximum of finish time.

$$\text{Makespan} = \max_{t_i \in T}(\text{FinishTime}_i) \quad (4)$$

Reducing makespan also reduces cost and increases workflow throughput.

3.2 Execution Cost

Cloud providers generally follow a pay-as-you-use pricing model. Therefore, workflow execution cost depends upon the processing time consumed on allocated virtual machines. The computation cost has been calculated using Eq. (5). The detailed explanation of the cost calculation is available in [35]-[38] researches.

$$\text{cost} = \frac{\text{CF}+\text{MF}}{2} \quad (5)$$

Where, CF is cost factor and MF is migration factor explained in [35]-[38] researches.

3.3 Degree of Imbalance

The degree of imbalance (DI) is used to find the load distribution on VMs. The formula of DI is given in Eq. (6).

$$\text{DI} = (T_{max} - T_{min})/T_{avg} \quad (6)$$

Where, T_{max}, T_{min}, and T_{avg} represent maximum, minimum and average loads respectively among all VMs.

3.4 Optimization Function

The workflow scheduling problem is formulated as a multi-objective optimization problem. The proposed scheduler attempts to

- Minimize workflow makespan,
- Minimize execution cost,
- Minimize degree of imbalance
- Maintain workflow precedence constraints.

These objectives are combined into a single fitness function (see Eq. (7)) using weighted normalization, $w_1 + w_2 + w_3 = 1$ [38]. The values of w_1, w_2, and w_3 are 0.5, 0.3 and 0.2 respectively.

$$\text{fitness} = \frac{w_1 \times \text{makespan} + w_2 \times \text{cost} + w_3 \times \text{DI}}{\text{DC} \times \text{VMs}} \quad (7)$$

Where, DC represents number of datacenter and VMs represents number of virtual machines.
The scheduler searches for the task-to-virtual-machine mapping that minimizes the above fitness value while preserving all workflow dependencies.

4 Proposed Methodology

This section explains the proposed method in detail. As discussed in introduction section, we have designed Elite PSO (EPSO) algorithm for workflows scheduling optimization. We have used three strategies along with HEFT [35]-[38] to improve the traditional PSO. The HEFT algorithm has been used to initialize the virtual machines with tasks ranking generated by HEFT. The HEFT algorithm is shown in Algorithm 1. Rest, other strategies are discussed one by one.

Strategy 1: In strategy one, we have used self-motivated inertia weight as discussed in [35]-[38]. This exponential decay based inertia improves the balance between exploration and exploitation of PSO. The SMIW strategy is represented in Eq. 8.

$$\omega = \omega_{max} \times \exp\left(-x \times \left(\frac{itr}{itr_{max}}\right)^y\right) \quad (8)$$

Where, ω_{max} is the initial inertia weight and it is set to 2.0. x is the local searching attractor and y is the global searching attractor.

Strategy 2: In strategy two, we have used adaptive tuning of cognitive and social parameters c_1 and c_2 which is represented by Eq. (9) and (10).

$$c_1(t) = c_1^{max} - (c_1^{max} - c_1^{min})\frac{t}{T_{max}} \quad (9)$$

$$c_2(t) = c_2^{min} - (c_2^{max} - c_2^{min})\frac{t}{T_{max}} \quad (10)$$

Where, t represents the current generation and T_{max} denotes the maximum number of generations.
In this research, c_1 is decreased linearly from 2.5 to 0.5 and c_2 is increased linearly from 0.5 to 2.5. This adaptive strategy exercises improved exploration and exploitation during earlier and later stages respectively and improves the convergence of the algorithm after combination of the strategy 3 which is explained next.

Algorithm 1: HEFT Algorithm

Input: Workloads T (Workflows or Independent Tasks), VMs V
Output: Initial Tasks Assignment to VMs

```
For each task t in T Do
    avg ← average(ET[t, vm] for all vm in V)
    If workflow Then
        rank(t) ← avg + max(rank(successors(t)))
    Else
        rank(t) ← avg
    End If
End For
Sort the tasks in descending order //maintain critical task priory
For each solution S Do
    Set all VMs v finish times = 0
    For each task t in sorted list Do
        For each vm in V Do
            Est ← max(finish time of predecessors)
            ft ← Est + ET[t, vm]
        End For
        Choose vm with minimum finish time ft
        Assign task t to vm and update finish time ft
    End For
End For
return initialized population (initial tasks assignments to VMs)
```

Strategy 3: This strategy preserves top 5% best solutions found of the population and guides the EPSO algorithm to move towards best scheduling solutions. The pseudocode of this strategy is demonstrated in Algorithm 2.

Algorithm 2: Proposed Elite Local Search

Input: Global Best Solutions (gbest), number of local search trials, VM List
Output: Improved Best Solution (gBest)

```
For k=1 to LS_Tries Do
    Randomly select a task dimension d
    Store the current VM assignment as oldVm
    Randomly select a new VM (newVM) such that newVM ≠ oldVM
    Temporarily assign task d to newVM
    Evaluate the fitness of the modified solution
    If new fitness < current fitness Then
        Accept the new assignment and update gBest fitness
    Else
        Restore the original VM assignment (oldVM)
    End If
End For
Return improved best schedule (gBest)
```

4.1 Proposed Algorithm

In this research we have fused all strategies mentioned above with PSO algorithm. The newly designed algorithm is called EPSO.

The mathematical model of the PSO algorithm (J. Kennedy and R. Eberhart, 1995) is presenting in Eqs. (11) and (12). Eq. (11) shows how to update the velocity of particles and Eq. (12) shows particles position update formula.

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (pBest_i - x_i^t) + c_2 r_2 (gBest - x_i^t) \quad (11)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (12)$$

Where, ω is inertia weight, c_1 is cognitive coefficient, c_2 is social coefficient, r_1 and r_2 are random numbers in the range (0, 1), x_i is current solution's position and v_i is current velocity.

Algorithm 3: Proposed EPSO Algorithm

Input: Generate Populations pop by HEFT policy, c_1, c_2, r_1, r_2, max_Generations etc.
Output: Best Schedule gBest

```
1. For Gen = 0 to max_Generations Do
2.     Compute SMIW inertia weight ω using Eq. (8)
3.     Compute c1(t) using Eq. (9)
4.     Compute c2(t) using Eq. (10)
5.     For each particle in pop Do
6.         For each dimension d Do
7.             Update velocity using Eq. (11) // using SMIW, c1(t) and c2(t)
8.             Update Position using Eq. (12)
9.         End For
10.        Evaluate fitness
11.        update pBest and gBest values
12.    End For
13.    Call Algorithm 2  // Preserving Best Solutions
14. End For
15. return Best Schedule (gBest)
```

Algorithm 3 is representing all steps involved in newly proposed EPSO algorithm. In algorithm 3, first of all population generated by HEFT algorithm is fed. The adaptive c1 and c2 are calculated for exploration in early stages and strong exploitation in later generations. The SMIW strategy given in Eq. 8 balances the exploration and exploitation phases. The velocity and positions are updated using Eqs. (11) and (12). In next step, Elite strategy mentioned in Algorithm 2 is called to guide the PSO for searching effective solutions using elite class preserved solutions. This process continues until the stopping condition is met.

5 Simulation Setup and Results Discussion

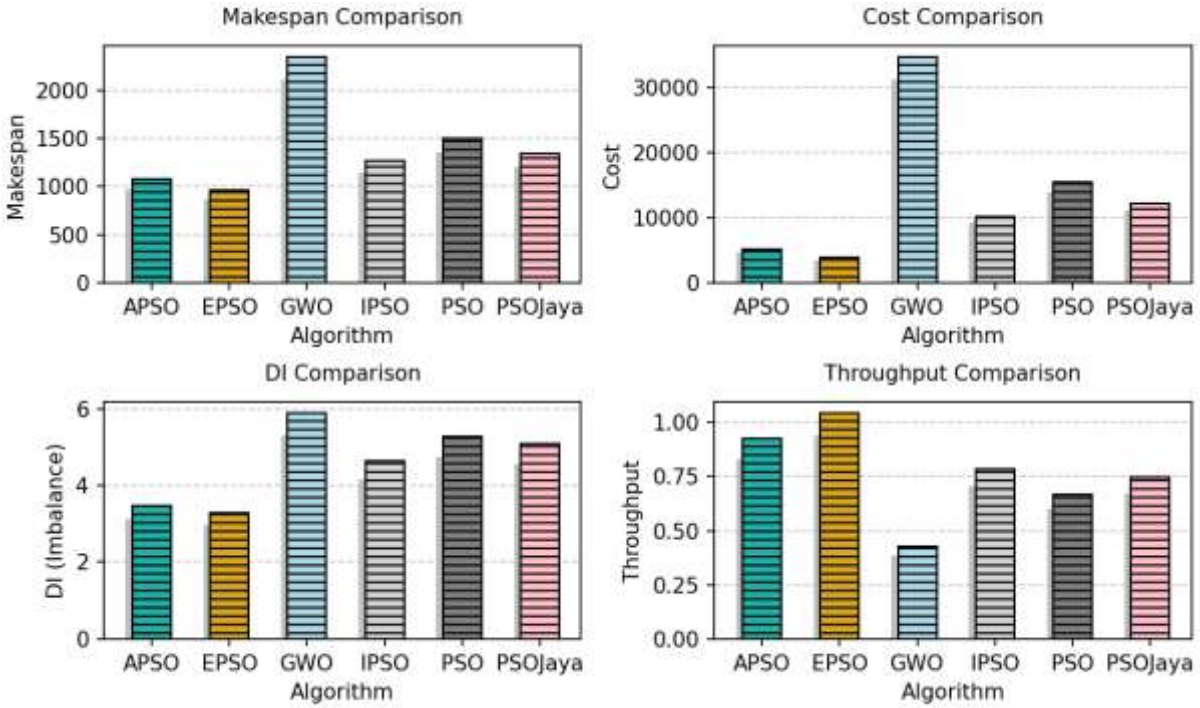
The simulation has been performed on a computing machine having processor 12th Gen Intel(R) Core(TM) i5-1235U (1.30 GHz), RAM 16.0 GB and OS Windows 11 Pro Edition. All algorithms were implemented in CloudSim 3.0 using Java 8 in NeatBeans to ensure fair experimental evaluation. A single datacenter comparison one host machine is considered in this research. A total of 50 virtual machines (VMs) were created with computing capacities ranging from 500-2500 MIPS, RAM capacities ranging from 512-2556 MB, bandwidth of 1000 Mbps, and one processing element (PE) per VM. The algorithms properties are given in Table 1.

Table 1: Algorithms Parameters

Algorithm Name	Parameter	Value
PSO, IPSO, APSO [38], PSOJaya and Proposed EPSO	Population (Particles)	100
	Max Generations	250
	c1, c2, r1 and r2	1.5, 1.5, r1 and r2 random range in (0, 1) respectively
GWO [27]	Population (Wolves)	100
	Max Generations	250

5.1 Results Discussion

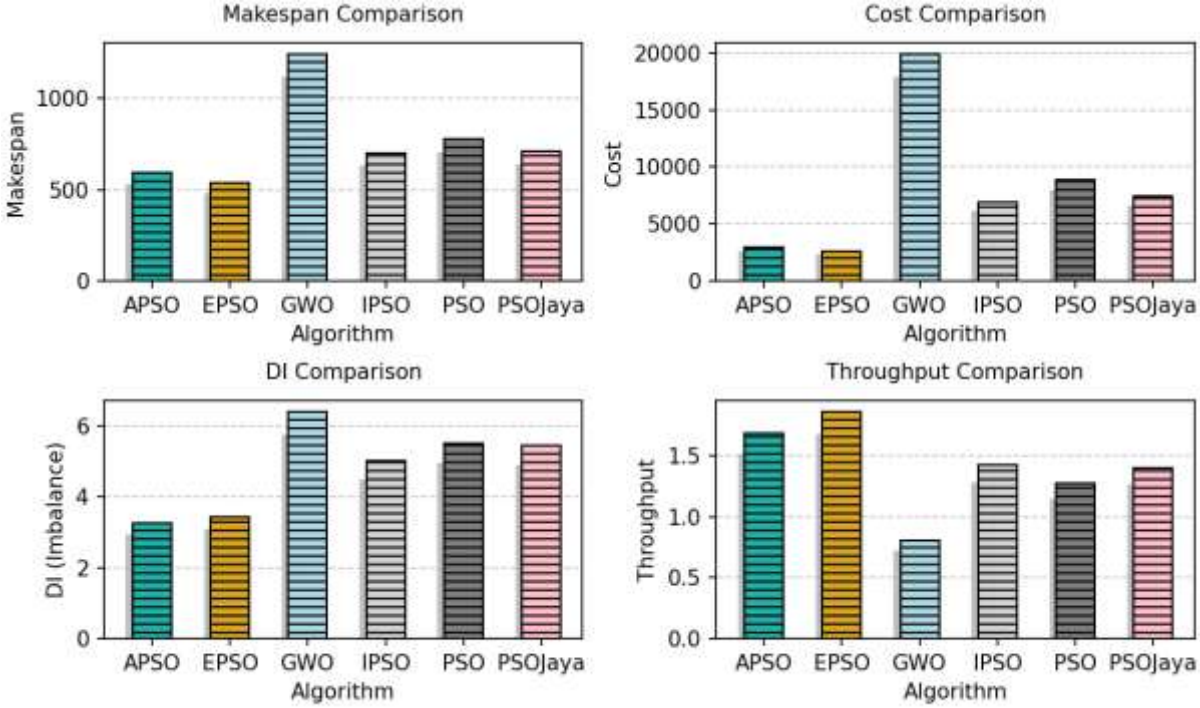
This section is presenting the results such as the average makespan, DI (degree of imbalance), cost and throughput obtained by all algorithms tested (20 times) on Workflows CyberShake, Montage, Epigenomics and Inspiral.



CyberShake Workload-1000

Figure 1: Results Using CyberShake Workload

Figure 1 compares the performance of proposed EPSO, APSO [38], IPSO, PSOJaya, PSO and GWO [27] algorithms using CyberShake-1000 workflow. Among all algorithms, the proposed EPSO algorithm consistently achieves the best overall performance. It records lowest makespan and faster workflow completion compared to its competitor APSO algorithm. The proposed EPSO algorithm also produces minimum execution cost, demonstrating efficient resources utilization. The EPSO attains lower degree of imbalance (DI) to APSO, proving a balanced distribution of tasks across virtual machines. Further, EPSO gives maximum throughput compared to other algorithms. This is due to elitism guidance and better load distribution.

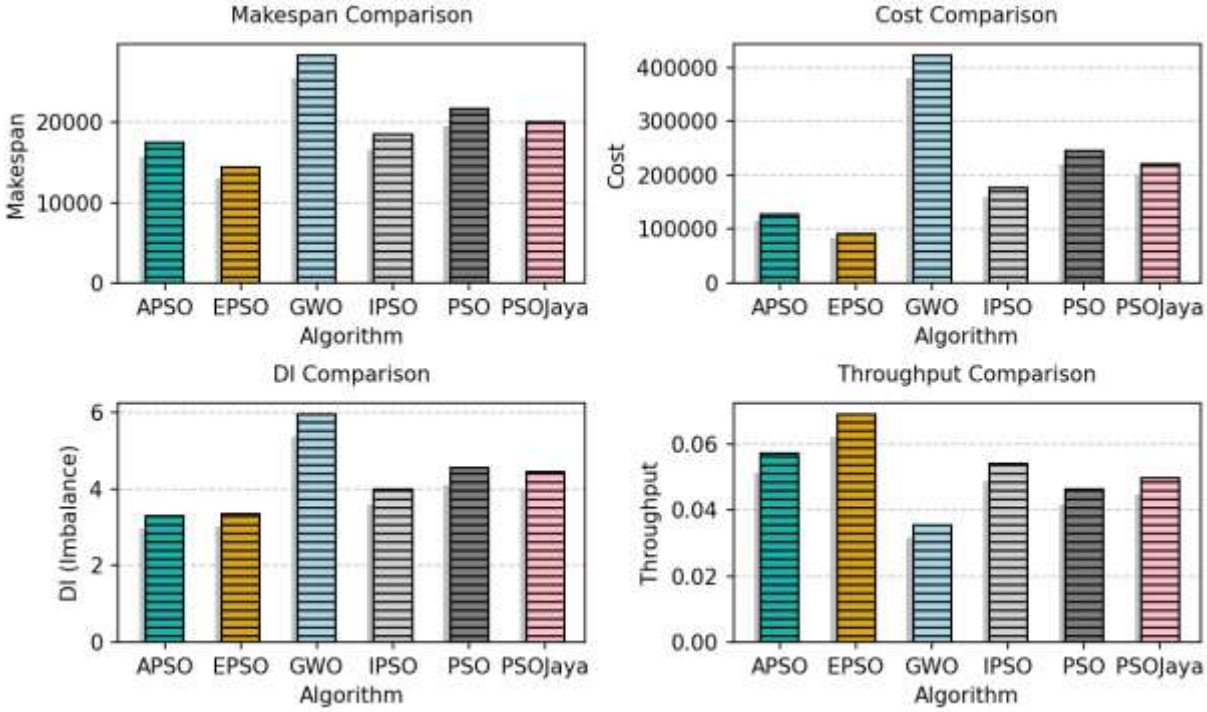


Montage Workload-1000

Figure 2: Results Using Montage Workload

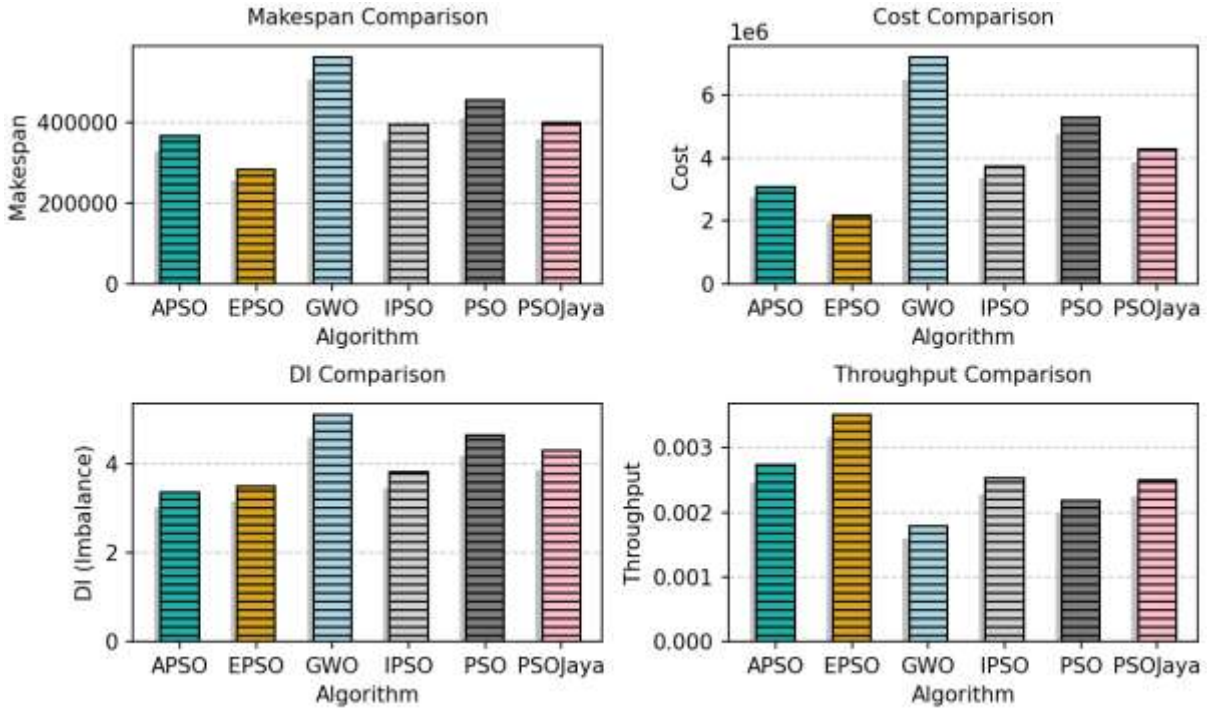
Figure 2 presents the performance comparison of proposed EPSO, APSO [38], IPSO, PSOJaya, PSO and GWO [27] methods using Montage-1000 workflow. The proposed EPSO demonstrates the best overall performance in

terms of lowest makespan, lowest cost, comparable DI to APSO and higher throughput, indicating faster workflow execution and better resource utilization. The proposed HEFT-Guided Elite PSO (EPSO) algorithm maintains good load distribution among all VMs in case of Montage data too. In contrast, GWO exhibits very poor performance compared to all algorithms. The elite PSO provides best performance due to faster convergence, good resource utilization and exploration. The second best algorithm is found APSO, which is previously [38] proposed by us.



Inspiral Workload-1000

Figure 3: Results Using Inspiral Workload



Epigenomics Workload-1000

Figure 4: Results Using Epigenomics Workload

Figure 3 compares the performance of the proposed EPSO, APSO [38], GWO [27], IPSO, PSO and PSOJaya algorithms on the Inspiral-1000 workflow. The proposed EPSO (Elite PSO) gets the best overall results, recording the lowest makespan, minimum execution cost and higher throughput. It shows the ability of EPSO to execute Inspiral workflows faster while utilizing cloud resources effectively. The EPSO shows comparable degree of imbalance to APSO [38], that shows effective load distribution among virtual machines in case of Inspiral data. The GWO performs worst in this research due to poor exploration and getting stuck in local optima. The elite learning mechanism in proposed EPSO algorithm improves convergence and scheduling quality for large scale workflow data.

Figure 4 illustrates the performance comparison of proposed Elite PSO (EPSO), APSO [38], GWO [27], PSO, IPSO, and PSOJaya on Epigenomics-1000 workflow data. The proposed EPSO consistently deliver the best performance in case of makespan, cost, and throughput. The degree of balance achieved by EPSO is comparable to APSO that means the EPSO distribute the load among all VMs efficiently. The GWO is proved worst algorithm in this experiment too whereas the best algorithm is EPSO.

The EPSO algorithm is working better due to its elitism property and better exploration and exploitation compared to other algorithms. It achieves faster convergence and all workflows are completed faster compared to its competitor APSO [38].

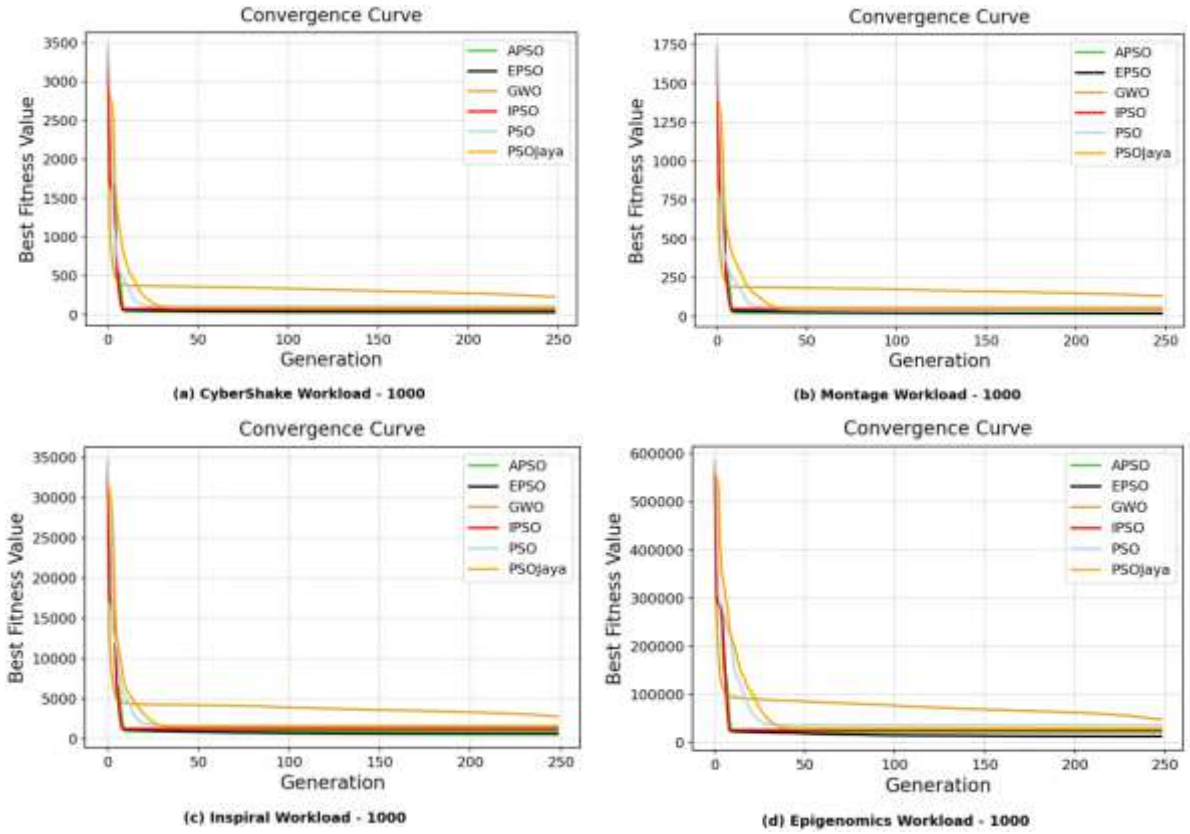


Figure 5: Convergence Analysis of All Algorithms

Figure 5 illustrate the convergence behavior of proposed EPSO, APSO, GWO, IPSO, PSOJaya, and PSO algorithm across all workflows. It is evident that the proposed EPSO method converges more rapidly than its competing algorithms. It reaches lower fitness values within early generations and maintains stability thereafter. In contrast PSO, IPSO, PSOJaya and especially GWO converge more slowly and remain at higher fitness values throughout the entire optimization process. The convergence behavior of EPSO demonstrates that the elite preservation policy effectively guides the swarm towards high class scheduling solutions. It prevents the EPSO to get stuck in local optima i.e. premature convergence across all workflows datasets.

6 Conclusion and Future Scopes

This paper presented a HEFT-Guided Elite Particle Swarm Optimization (EPSO) algorithm for workflows scheduling in heterogeneous cloud computing environments. The proposed method combines the HEFT policy for VMs initialization, adaptive strategy for better exploration and exploitation and elite solution preservation

mechanism to search better scheduling solution. The effectiveness of the proposed EPSO algorithm was tested suing four widely adopted scientific workflow datasets namely CyberShake, Montage, Inspiral and Epigenomics. Comparative experiments against APSO, IPSO, GWO and PSOJaya demonstrated that EPSO algorithm consistently achieved lower makespan, reduced execution cost, improved load distribution and throughput. Furthermore, the convergence analysis confirmed that the proposed elite guided search strategies accelerates the convergence while avoiding premature convergence and guides the proposed algorithm to search high quality solutions within few initial generations. Overall, the experimental findings indicate that adaptive particle swarm optimization with elite class guidance enhances the workflows scheduling performance in heterogeneous cloud environment. The proposed EPSO algorithm represents an effective and computationally efficient approach for executing large scale workflows. Compared to APSO [38], proposed EPSO algorithm improves the makespan by 11% and cost by 25% for CyberShake dataset, the makespan by 9% and cost by 9% for Montage dataset, the makespan by 17% and cost by 28% for Inspiral dataset and the makespan by 22% and cost by 29% for Epigenomics dataset.

In future, more experiments can be conducted with other dataset like Google Cluster 2019 traces and other independent tasks. The proposed algorithm can be test in other engineering areas as well. A new strategy can be proposed to enhance the overall cloud performance and it can be applied on datacenter level.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging It Platforms: Vision, Hype, and Reality for Delivering Computing as The 5th Utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, 2009.

[2] M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.

[3] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," Journal of Grid Computing, vol. 3, nos. 3–4, pp. 171–200, 2005.

[4] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. New York, NY, USA: W. H. Freeman, 1979.

[5] 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, Mar. 2002.

[6] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," in Proc. 24th IEEE Int. Conf. Advanced Information Networking and Applications (AINA), Perth, Australia, 2010, pp. 400–407.

[7] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in Proc. IEEE Int. Conf. Neural Networks, Perth, Australia, 1995, pp. 1942–1948.

[8] Y. Shi and R. Eberhart, "A Modified Particle Swarm Optimizer," in Proc. IEEE Int. Conf. Evolutionary Computation, Anchorage, AK, USA, 1998, pp. 69–73.

[9] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer with Time-Varying Acceleration Coefficients," IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 240–255, 2004.

[10] M. Clerc and J. Kennedy, "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space," IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58–73, 2002.

[11] R. Poli, J. Kennedy, and T. Blackwell, "Particle Swarm Optimization: an Overview," Swarm Intelligence, vol. 1, no. 1, pp. 33–57, 2007.

[12] H. Arabnejad and J. Barbosa, "List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp. 682–694, 2014.

[13] L. F. Bittencourt, R. Sakellariou, and E. R. M. Madeira, "Dag Scheduling using a Lookahead Variant of The Heterogeneous Earliest Finish Time Algorithm," Journal of Parallel and Distributed Computing, vol. 70, no. 10, pp. 984–996, 2010.

[14] R. Sakellariou and H. Zhao, "A Hybrid Heuristic for Dag Scheduling on Heterogeneous Systems," in Proc. IEEE Int. Parallel and Distributed Processing Symp. (IPDPS), 2004.

[15] A. Verma and S. Kaushal, "Cost-time Efficient Scheduling Plan for Executing Workflows in The Cloud," Journal of Grid Computing, vol. 13, no. 4, pp. 495–506, 2015.

[16] R. Prodan and M. Wieczorek, "Bi-Criteria Scheduling of Scientific Grid Workflows," IEEE Transactions on Automation Science and Engineering, vol. 7, no. 2, pp. 364–376, 2010.

[17] E. Deelman et al., "Pegasus: a Workflow Management System for Science Automation," Future Generation Computer Systems, vol. 46, pp. 17–35, 2015.

[18] W. Chen, E. Deelman, and T. Fahringer, "Dynamic Workflow Scheduling: a Survey," Future Generation Computer Systems, vol. 110, pp. 686–699, 2020.
[19] X. Liu, Z. Li, and H. Wang, "Adaptive Particle Swarm Optimization for Workflow Scheduling in Cloud Computing," Cluster Computing, vol. 24, no. 2, pp. 1381–1396, 2021.
[20] M. A. Tawfeek, A. El-Sisi, A. Keshk, and F. Torkey, "Cloud Task Scheduling Based on Ant Colony Optimization," International Arab Journal of Information Technology, vol. 12, no. 2, pp. 129–137, 2015.
[21] K. Rajakumari and S. Mala, "Fuzzy-Based Ant Colony Optimization Scheduling in Cloud Computing," Computer Systems Science and Engineering, vol. 40, no. 2, pp. 671–687, 2021.
[22] C. Chandrashekar et al., "Hybrid Weighted Ant Colony Optimization Algorithm for Cloud Task Scheduling," Applied Sciences, vol. 13, no. 6, Art. no. 3433, 2023.
[23] G. Li, G. Wu, D. Li, and S. Arunagiri, "An Improved Ant Colony Optimization Task Scheduling Algorithm," Future Internet, vol. 11, no. 4, 2019.
[24] X. Liu, Y. Zhang, and J. Chen, "Cloud Computing Task Scheduling Based on Improved PSO-ACO Algorithm," in Proc. ACM Int. Conf. Computing Technologies, 2024.
[25] X. Liu, Y. Zhang, and J. Chen, "Cloud Computing Task Scheduling Strategy Based on Improved Ant Colony Optimization Algorithm," 2024.
[26] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, 2014.
[27] S. Khaleelahmed, S. Selvaraj, R. B. Mohite, M. L. Bangare, P. M. Bangare, S. S. Kulkarni, S. M. Ajibade, and A Raghuvanshi, "Task Scheduling Algorithm Using Grey Wolf Optimization Technique in Cloud Computing Environment," Bulletin of Electrical Engineering and Informatics, vol. 14, no. 4, 2762-2771, 2025.
[28] P. Singh, R. Sharma, and K. Verma, "Hybrid GWO-PSO Based Task Scheduling for Cloud Computing," International Journal of Intelligent Systems, vol. 35, no. 11, pp. 1720–1745, 2020.
[29] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent Advancements in Resource Allocation Techniques for Cloud Computing Environment: A Systematic Review," Cluster Computing, vol. 20, no. 3, pp. 2489–2533, 2017.
[30] H. Khurana and Y. Singh, "Hybrid Heuristic-Based Workflow Scheduling in Cloud Computing: A Survey," Engineering Applications of Artificial Intelligence, vol. 82, pp. 288–303, 2019.
[31] M. Rodríguez and R. Buyya, "Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds," IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 222–235, 2014.
[32] L. Abualigah et al., "Recent Advances in Nature-Inspired Optimization Algorithms for Engineering Applications: A Survey," Archives of Computational Methods in Engineering, vol. 29, pp. 2281–2332, 2022.
[33] Z. Beheshti and S. M. H. Shamsuddin, "A Review of Population-Based Metaheuristic Algorithms," International Journal of Advances in Soft Computing and Its Applications, vol. 5, no. 1, pp. 1–35, 2013.
[34] S. Mirjalili, "Evolutionary Algorithms and Neural Networks: Theory and Applications," Springer, 2019.
[35] J. Bhagwan, and S. Kumar, “An HC-CSO Algorithm for Workflow Scheduling in Heterogeneous Cloud Computing System,” International Journal of Advanced Computer Science and Applications, vol. 12, no. 6, pp. 484-492, 2021.
[36] J. Bhagwan, and S. Kumar, “An H-CSO Algorithm for Workflow Scheduling in Heterogeneous Cloud Environment,” International Journal of Intelligent Engineering & Systems, vol. 14, no. 5, pp. 422-434, 2021.
[37] J. Bhagwan, S. Rani, Manoj, S. Godara, Yashasvi, and S. Kumar, “IJPSO: An Improved Hybrid PSO Algorithm for Multi-Type and Large Scale Data Scheduling In Cloud Computing, International Journal of Artificial Intelligence and Machine Learning, vol. 6, no. 3s, pp. 399-412, 2026.
[38] A. Bishnoi, J. Bhagwan and S. Kumar, “Adaptive PSO Algorithm for Resource Allocation to Large Scale Data in Heterogeneous Cloud Computing Environment,” International Journal of Artificial Intelligence and Machine Learning, vol. 6, no. 3s, pp. 736-747, 2026.