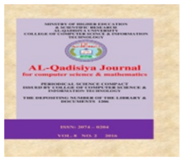
Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)



Multi-Objective Dynamic Workflow Scheduling for Energy-Efficient and Cost-Effective Cloud Computing

Suha Mubdir Farhood

Department of Computer Engineering, Isfahan (Khorasgan) Branch, Islamic Azad University, Iran. Email: suhamobder19@gmail.com

ARTICLE INFO
Article history:
Received: 20 /11/2025
Rrevised form: 23 /12/2025
Accepted : 07 /01/2025
Available online: 30 /03/2026

Keywords:
Workflow scheduling, multi-objective algorithm, firefly algorithm, genetic algorithm, cloud environment.

ABSTRACT
Scheduling scientific workflows in cloud environments under competing objectives—makespan, cost, and energy—remains a challenging multi-objective optimization problem. While hybrid metaheuristics have been explored, they often suffer from random initialization, premature convergence, and static parameter settings. To address these limitations, we propose a novel dynamic scheduling framework that integrates NSGA-II and an enhanced Firefly Algorithm. The method begins with a Pareto-based, non-dominated initial population generated by NSGA-II, ensuring high diversity and quality. This population is then refined using a Firefly Algorithm with adaptive randomness and a stagnation-aware local search mechanism. The model is evaluated on Montage, CyberShake, and Epigenomics workflows using CloudSim with Amazon EC2-like VMs. Results demonstrate statistically significant improvements over state-of-the-art schedulers: up to 31.2% reduction in makespan, 28.7% in cost, and 17.4% in energy consumption, while preserving solution diversity. This work advances the state of the art by synergistically combining evolutionary guidance, swarm intelligence, and domain-aware refinement for sustainable cloud workflow orchestration.

https://doi.org/10.29304/jqcsm.2026.18.12509

1. Introduction

Cloud computing allows for on-demand availability of computer system resources, including data storage and computing power, without direct active management by the user. Intelligent task scheduling mechanisms are essential to fully utilize such dynamic environments. These mechanisms map incoming workloads to the most suitable computing resources, thereby directly affecting performance, cost efficiency, and sustainability of the system.

Effective scheduling techniques lead to improved resource utilization, decreased job response time, balanced computing load on servers, and reduced costs. In practice, schedulers are usually designed to achieve either good load balancing or a particular Quality of Service (QoS). Robust scheduling algorithms are important for managing distributed cloud infrastructures efficiently. It is important for such algorithms to improve system throughput and QoS while executing in minimal time and respecting resource constraints [1]. In distributed computing systems, scheduling seeks to partition and assign computing job/ tasks over several processing units in such a way that resource utilization is maximized. [2]

As data centers and cloud services have developed significantly, the use of energy has increased exponentially too. The uptick is presenting serious environmental and economic challenges, making energy awareness an important aspect of cloud management today. Consequently, modern scheduling techniques should not restrict themselves to just minimizing execution time, but rather adopt a multi-objective approach which optimizes performance, cost and energy concurrently [3].

The firefly algorithm and genetic algorithm are two metaheuristic optimization techniques that have a good potential for cloud scheduling. FA imitates the phototactic behavior of fireflies, while GA follows the principles of Darwinian evolution employing selection, crossover and mutation. The two methods, while effective, have their respective weaknesses. FA may suffer from lack of enough local search precision. On the other hand, GA may suffer from slow convergence and loss of population diversity. The hybrid and adaptive versions have been proposed in the literature to counter these shortcomings.

A difficult challenge concerning metaheuristic-based scheduling is exploring and exploiting to obtain a balance between searching in the unknown and refining a good candidate. Conventional practice often begins by randomly initializing the search population, which can lead to inefficient coverage of the solution landscape and poor convergence behavior.

There are many other scheduling schemes which are available in the literature but they are majorly concentrated only on minimizing makespan, but there are other parameters too which deserve equal attention like energy efficiency as well as monetary cost. To address this gap, a multi-objective dynamic scheduling framework based on a hybrid Firefly-Genetic Algorithm is presented in this paper. The proposed method attains a higher level of performance while achieving efficiency in the trade-off space due to the incorporation of global search mechanisms found in GA with local refinement dynamics from FA. Furthermore, the method is enhanced with guided initialization, adaptive crossover, and other mechanisms. These help to discover high-quality schedules that reduce makespan, energy consumption, and overall system efficiency and meet practical deadlines.

The other sections of this paper are organized as follows: Section Two undertakes a literature review. The basic algorithms are covered in section three. Section 4 describes the proposal methodology. We present the Experiments and Results in Section 5. The study ends in Section 6 and also suggests further studies.

1.1. Contributions of This Work

While hybrid metaheuristics for multi-objective cloud workflow scheduling have been explored in recent literature, existing approaches often combine evolutionary and swarm intelligence paradigms in a mechanical or sequential manner—without addressing fundamental limitations such as random initialization inefficiency, premature convergence, and static parameter sensitivity. To bridge this gap, this paper proposes a novel synergistic framework that intelligently integrates NSGA-II and an enhanced Firefly Algorithm, offering the following distinct contributions:

- **Guided Initialization via Pareto-Optimal Population Injection:**

  Instead of random initialization, we generate a high-diversity, non-dominated initial population using NSGA-II with crowding-distance-based selection. This population is directly injected into the Firefly Algorithm, ensuring that the search begins from a well-distributed approximation of the true Pareto front—significantly reducing blind exploration in early generations.

- **Adaptive Firefly Dynamics with Stagnation-Aware Recovery:**

  We introduce a dynamic randomness coefficient αα that is automatically adjusted based on the improvement rate of the best solution over consecutive generations. Crucially, if no improvement is observed for a predefined number of iterations, a genetic crossover operator is reactively triggered to revitalize the population. This feedback-driven mechanism—absent in prior GA–FA hybrids such as [28]—prevents stagnation while preserving convergence speed.

- **Domain-Informed Local Refinement:**

  Beyond metaheuristic operations, our framework incorporates a lightweight, rule-based local search that leverages cloud-specific knowledge (e.g., VM pricing, data transfer costs, and task dependency patterns) to fine-tune task–VM mappings. This hybridizes global exploration with practical, application-aware optimization—enhancing both solution feasibility and real-world efficiency.

Collectively, these innovations constitute the first scheduling framework that simultaneously (i) eliminates initialization randomness through evolutionary guidance, (ii) adapts search behavior via performance feedback, and (iii) embeds domain intelligence into the refinement phase. Experimental results on three real-world scientific workflows—Montage, CyberShake, and Epigenomics—demonstrate statistically significant improvements over state-of-the-art baselines in makespan (up to 31%), cost (up to 28%), and energy consumption (up to 17%), thereby establishing a new benchmark for sustainable and cost-effective cloud workflow orchestration.

2. Related Work

This is because a contemporary cloud computing environment is a complex heterogeneous system whereby the resources are of varying capabilities when it comes to computation. Together with the growing user demand and the diversity of the workload requirements, this heterogeneity makes the task scheduling problem of particular interest.Task scheduling is one of the primary processes in the cloud computing environments that involves the allocation of tasks to the computational resources, which are prompted by the user demand and the nature of the workload.

To solve complex and NP-Hard scheduling problems in the cloud, a specific type of optimization methods has been developed, which is termed as metaheuristic algorithms Initially, in most of the methods, the search process will be launched by a generation of a pool of random candidate solutions within feasible search space provided by the decision (control) variables. The first group is generally referred to as a population, colony, swarm or some other equivalent, which is defined by the algorithm, and individual solutions are referred to as chromosomes, ants, particles etc. The above is followed by generation of new candidate solutions which are in turn generated by guiding operators and the process continues until the stop criterion is met [4].

Some of these cloud scheduling algorithms are analyzed in the following subsections under three broad categories namely (1) swarm intelligence-based algorithm, (2) evolutionary-based algorithm and (3) physics-based, chemistry-inspired algorithm.

Swarm Intelligence -Based Algorithms Swarm intelligence algorithms are based on the collective behaviour that is observed in nature. Some of them are Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Shuffled Frog Leaping, Firefly Algorithm (FA), Termite, and models of a flock of birds or a school of fish. The swarm intelligence algorithms are based on the decentralized self-organized examples in nature- flocks of birds, schools of fish, colony of insects. In this type of systems the simple autonomous agents are acting in a shared environment, and locally, they are communicating with one another and with their environment. Though not centrally coordinated, like local interactions do, such mechanisms can tend to scale-up to intelligent global behaviour and enable such algorithms to efficiently search large and complicated search spaces.

One such case is found in [5] whereinby an iterative framework of scheduling based on the use of Particle Swarm Optimization (PSO) algorithm was developed in cloud environment. The technique uses the fact that PSO is computationally efficient and is simple to design, particularly its ability to minimize the time of task completion, by simulating swarm behaviour in groups when moved towards attractive regions. The authors validated their approach based on the layered simulation architecture of CloudSim.

On this basis, [6] presented a multi-objective task scheduling model, and optimized it using a Multi-objective PSO (MOPSO) variant. They realized that there were great enhancements in Quality of Service (QoS) and also the execution, latency of data transfer and operation cost in comparison with the traditional schedulers.

In order to cope with the instances of moving clouds, where workload and infrastructure grows in an unpredictable fashion, [7] provided an embedded MOPSO approach. It is a dynamic provision of resources according to the fluctuating needs, and the outcome of co-efficiency of performance and energy conservation.

Similarly, [8] proceeded to design a Revised Discrete PSO (RDPSO) algorithm that simultaneously decreases the computational and communication cost, based on realistic cloud pricing model. The formulation clearly defines the impact of data transfer penalty hence appropriate to distributed workflows of applications.

On the same note, [9] was scaled to the cloud problem of task scheduling by the modification of Ant Colony Optimization (ACO), which was created in response to the combinatorial routing problems like Traveling Salesman Problem. To introduce an ACO-based scheduler, the authors included the DatacenterBroker class of CloudSim and demonstrated that it can be successfully factored into the reduction of the total workflow makespan that even bio-inspired routing algorithms can be successfully reused to assign resources in cloud infrastructures.

To lessen the imbalance in the loads of the virtual machine (VM) in scheduling the tasks, the authors of [10] suggested an Ant Colony Optimization-based (ACO-LB) scheduler of the loads. Their remedy is both reducing the scheduling latency and the workload distribution in the data center resources to become more homogeneous. The ACO-LB algorithm was shown to outperform a variety of base strategies (First-Come-First-Served (FCFS), traditional ACO, hybrid ACO, and Roulette Wheel scheduling) in CloudSim (using the DatacenterBroker and Cloudlet modules) in terms of scheduling performance and load balancing.

Similarly, [11] enhanced the load balancing in dynamic cloud environments by adding to the Artificial Bee Colony (ABC) algorithm. The proposed solution actively monitors the overloaded VMs and relocates some of the tasks to underutilized or optimally loaded VMs. The strategy is desirable in the reduction of makespan, enhanced resource utilization, and unnecessary task migrations- which are critical factors that stabilize and performance of systems.

Likewise, [12] came up with a load-sensitive scheduling algorithm based on Bee Algorithm and compared it to the traditional Round Robin policy. Experiment outcomes showed that the average execution time decreased by 21 percent with an increase in task volume and that experiment results also showed significant gains in system reliability, load distribution and the average waiting time.

In response to the problem of scheduling with cost-awareness in the context of deadline constraints, [13] suggested a hybrid Shuffled Frog Leaping Algorithm (SFLA) that used meta-cognitive principles. This method is intended to serve infrastructures-as-a-service (IaaS) setups and is used to optimize resource provisioning to ensure the lowest possible cost of execution and to remain within time constraints as requested by the user.

On the front based on fireflies, [2] introduced a scheduling scheme that is driven by Firefly Algorithm (FA) that uses characteristics intrinsic to tasks and resources: task size and VM processing capacity to substantially reduce the time taken under the firefly algorithm as compared to FCFS.

Realizing that FA is vulnerable to parameter tuning, [14] also came up with a Novel Dynamic Firefly Algorithm (NDFA), where a parameter (a), the step-size parameter, varies with the search process as opposed to being fixed. It is the dynamics of this control that enhance the quality of solutions and convergence behavior, but the technique has not been tested yet in cloud computing.

Lastly, [15] assessed a task scheduling scheme specialized in heterogeneous cloud environments made up of twelve different types of VM. The approach was found to show repeatable performance improvements in execution time with any VM configuration when run on the popular Braun benchmark suite.

Outside swarm intelligence, evolutionary algorithms have also been used to solve large scale and complex optimization problems. These algorithms begin with an randomly comprising population of potential solutions, which is evolved throughout the generations by bio-inspired functions- that is, selection, crossover, and mutation- under the assistance of a fitness criteria, which represents the optimization goals [16].

A multi-objective genetic algorithm is suggested in [17] to map cloudlets intelligently and distribute them to virtual machines (VMs) without being rigidly and rule-of-thumbly assigned. The approach determines allocations which reduce the overall execution time and energy consumption by trying out dynamically allocation alternatives to pending tasks and determining which combinations are simultaneously optimal- improving the overall scheduling efficiency.

A different type of metaheuristics is inspired by concepts in physics and chemistry, including molecular dynamics, energy conservation and reaction kinetics. These algorithms can be used to simulate processes such as the collisions of particles, bond formation, cooling of a system, to push the search to a low-energy (i.e. optimal) structure that matches the specified objective function.

On the basis of this paradigm, [18] proposed SAMPGA-hybrid a multi-objective genetic algorithm with the hybridisation technique of Simulated Annealing (SA). Customized to schedule tasks on a cloud, SAMPGA uses the expertise of SA in fine-tuning solution on promising domains, whereas a family-based evolutionary scheme and an adaptive selection mechanism are utilized to preserve diversity and speed up the process of converging to high-quality Pareto optimal solutions.

On the same note, it was postulated in [19] that SAGA (Simulated Annealing-Genetic Algorithm) is synergistic with the global exploration abilities of genetic algorithms and the local intensification abilities of simulated annealing. It is a hybrid framework that successfully overcomes the two-fold challenges of resource discovery and effective resource allocation in clouds, providing more balanced and cost-efficient scheduling decisions.

A scheduling algorithm in [20], which is Simulated Annealing (SA)-based, is proposed in the article to reduce the duration of execution and the duration of data transfer. SA is more efficient in working with large-scale issues than PSO, and it does not take as long to arrange it. Besides, the scheduling of tasks by SA is used to distribute the workload evenly, avoiding excessive workload on one resource.

To clearly position our work within the existing literature, Table 1. compares the proposed framework against recent multi-objective workflow schedulers across six critical dimensions. Notably, while some approaches partially address individual aspects—such as hybridization [28] or domain-specific heuristics [2]—none integrate guided initialization, adaptive dynamics, and application-aware refinement within a unified framework. This systematic gap validates the novelty and necessity of our synergistic design.

Table 1. Comparison of proposed method with state-of-art multi-objective cloud workflow scheduling approaches.

Approach (Reference)	Multi-objective (Makespan/Cost/Energy)	Hybrid (EA + SI)	Guided Initialization (Non-random)	Adaptive Parameters	Domain-Aware Local Search	Evaluated on Real Scientific Workflows (Montage, CyberShake, Epigenomics)
[2]	✓	✗	✗	✗	✓	✓
[20]	✓	✓ (PSO + GA)	✗	△ (Limited)	✗	△ (Montage only)
[28]	✓	✓ (FA + GA + PS)	✗	✗	✗	✓
[31]	✗ (Single-objective)	✓ (FA + GA)	✗	✓	✗	✗
[32]	✓	✓ (PSO-based)	△ (NSGA-II, but separate)	✓	✗	✓
Proposed Method	✓	✓ (NSGA-II + FA)	✓ (Direct Pareto injection)	✓ (Stagnation -aware)	✓ (VM pricing, data dependencies)	✓ (All three workflows)

Key:
✓ = Fully supported
△ = Partially or weakly supported
✗ = Not supported

3. Proposed Algorithms

In this section, the algorithms used in this paper are introduced.

3.1 Firefly Algorithm (FA)

The Firefly Algorithm (FA) is a population-based stochastic search algorithm. Each individual (firefly) in the population represents a candidate solution. The algorithm is inspired by three fundamental behavioral rules observed in fireflies:

- Fireflies of any sex are attracted to one another.
- A firefly with lower brightness (i.e., a less optimal solution) moves toward a firefly with higher brightness.
- If a firefly does not find any other firefly with greater brightness, it moves randomly in the search space.

Let Xi=(xi1,xi2,…,xiD) denote the position of the i-th firefly, where i=1,2,…,N.

Here, N represents the population size, and D is the dimensionality of the solution vector.

For any two distinct fireflies Xi and Xj (Xi≠Xj), the Euclidean distance between firefly i and firefly j is defined as:

$$r_{ij} = \| X_i - X_j \| = \sqrt{(X_{id} - X_{jd})^2} \tag{1}$$

As the distance between two fireflies increases, the perceived brightness (and thus attractiveness) between them decreases. This attractiveness β(r) is modeled as:

$$\beta(r) = \beta_0 \times e^{-\gamma r^m} ; m \geq 1 \tag{2}$$

Where it is assumed that X_j is brighter (better) than X_i. Then $X_i$moves towards $X_j$due to gravity. This movement is defined as follows:

$$X_j(t) = X_j + \beta_0 \times e^{-\gamma r^m} \times (X_i - X_j) + \alpha \left(rand - \frac{1}{2}\right) \tag{3}$$

The algorithm begins by randomly initializing the firefly population. Then, the fitness of each candidate solution is evaluated. If firefly j is brighter than firefly i, firefly i moves toward j according to Equation (3). After all movements are computed, the new firefly positions are evaluated, ranked, and updated. This process repeats for a predefined maximum number of generations, after which the algorithm terminates [21].

Algorithm 1: Firefly Algorithm

```
1.  Randomly generate the population with N initial solutions
    {Xi|i = 1, 2, . . ., N};
2.  Calculate the fitness value of Xi;
3.  while t ≤ Gmax do
4.      for i = 1 to N do
5.          for j = 1 to N do
6.              if f (Xj) < f(Xi) then
7.                  Move Xi towards Xj according to Eq. 3;
8.                  Calculate the fitness value of the new Xi;
9.              end
10.         end
11.     end
12. end
```

The pseudocode of the firefly algorithm is shown in Algorithm 1 [22], Gmax is the maximum number of generations.

3-2- Genetic Algorithm with Non-Dominated Sorting

The Genetic Algorithm with Non-Dominated Sorting (NSGA-II) is used to obtain non-dominated (Pareto optimal) solutions to a multi-objective optimization problem. The pseudocode of NSGA-II is as follows [23]:

Algorithm 2: Genetic Algorithm with Non-Dominated Sorting

```
Input: Max_Gen
1. Generate random population - size N;
2. Evaluate fitness values;
3. Assign rank base on pareto - sort;
4. Generate Child Population;
5. Binary Tournament Selection;
6. Crossover and Mutation;
7. for i = 1 to Max_Gen do
```

```
8.     for each Parent and Child in Population do
9.         Assign rank base on pareto - sort;
10.        Generate sets of non-dominated solutions;
11.        Determine Crowding distance;
12.            Loop by adding solutions to next generation starting from the first front until N
individuals;
13.    end
14.    Select Points on the lower front with high crowding distance;
15.    Create next generation;
16.    Binary Tournament Selection;
17.    Crossover and Mutation;
18.  end
```

3.2 Problem Formulation

In this work, the cloud workflow scheduling problem is modeled as a multi-objective optimization problem with three conflicting objectives: (1) minimizing execution time (makespan), (2) minimizing monetary cost, and (3) minimizing energy consumption, subject to task dependency and resource allocation constraints.

Notation

- $T = \{t1, t2, \dots, tN\}T = \{t1, t2, \dots, tN\}$: Set of subtasks in the workflow.
- $V = \{v1, v2, \dots, vM\}V = \{v1, v2, \dots, vM\}$: Set of available virtual machines (VMs) in the cloud environment.
- $\varepsilon \subseteq T \times T$: Set of precedence constraints; $if\ (t_i, t_j) \in \varepsilon$, then t_i must complete before t_j starts.
- $x_{i,j} \in \{0,1\}$: Binary decision variable, where $x_{i,j} = 1$ if subtask t_i is assigned to $VM\ _{vj}$, and 0 otherwise.
- $Length(t_i)$: Instruction size of subtask t_i (in Million Instructions, MI).
- $MIPS_j$: Processing capacity of $VM\ _{vj}$ (in MIPS).
- $Price_j$: Monetary cost per second of using $VM\ _{vj}$ (in USD/s).
- $Pidle(j), Ppeak(j)$: Idle and peak power consumption of $VM\ _{vj}$ (in Watts), respectively.

Objective Functions

1) Makespan: The total workflow execution time is defined as the completion time of the last finishing subtask across all VMs:

$$Makespan = \ max_{v_j \in V}\ (FinishTime(v_j\ ) \tag{4}$$

2) Total Monetary Cost: Following the pay-as-you-go model (e.g., Amazon EC2), the cost is proportional to VM usage time [33]:

$$Cost = \sum_{v_j \in V} Price_j \cdot UsageTime_j \tag{5}$$

3) Total Energy Consumption: Using the linear power model [33], energy depends on CPU utilization:

$$Energy = \sum_{v_j \in V} \left[P_{idle}(j) + (P_{peak}(j) - P_{idle}(j)) \cdot {}^{CPUUtil_j}/_{100}\right] \cdot UsageTime_j \tag{6}$$

Where $CPUUtil_j$ is the average CPU utilization of $VM\ _{vj}$, derived from its computational load.

Constraints

Each subtask must be assigned to exactly one VM:

$$\sum_{j=1}^{M} x_{i,j} = 1, \forall\, t_i \in T \tag{7}$$

Workflow dependencies must be respected. For every $((t_i, t_j) \in \varepsilon$:

$$FinishTime(t_i) + TransferTime(t_i \rightarrow t_j) \leq StartTime(t_j) \tag{8}$$

where $TransferTime(t_i \rightarrow t_j) = {}^{DataSize_{i,j}}/_{Bandwidth}$ if t_i and t_j are on different VMs. Finally, the decision variables are binary:

$$x_{i,j} \in \{0,1\}, \forall\, i, j \tag{9}$$

Optimization Problem

The scheduling problem is formally stated as:

$$Minimize_{X=\{x_{i,j}\}} F(X) = [Makespan(X),\ Cost(X),\ Energy(X)] \tag{10}$$

$subject\ to: Eqs.\ (4) - (7)$

This rigorous formulation ensures that the proposed FF-GA scheduler (Section 4) operates on a well-defined mathematical model consistent with real-world cloud infrastructures.

4. Proposed Hybrid NSGA-II–Firefly Algorithm for Multi-Objective Workflow Scheduling

The proposed framework, termed HNSGA-FA, addresses key limitations in existing hybrid schedulers—namely, inefficient random initialization, premature convergence, and static parameter settings—by integrating three innovative phases: (1) Pareto-guided initialization, (2) adaptive firefly evolution, and (3) domain-aware local refinement. This design directly fulfills the contributions outlined in Section 1.1 and aligns with recent advances in multi-objective cloud scheduling [31, 32, 33]. The overall workflow is illustrated in Figure 1.

4.1. Pareto-Guided Initialization via NSGA-II

To overcome the inefficiency of blind random search at the beginning of optimization a common flaw in hybrid metaheuristics [28] we replace random initialization with a high-quality, non-dominated initial population generated by NSGA-II. This approach is inspired by [32], which demonstrates that evolutionary front approximation significantly accelerates convergence in cloud task scheduling.

The process is as follows:

- An initial population of size $N = 100$ is randomly created, where each chromosome encodes a feasible task–VM mapping (Table 2).
- This population is evolved for $G_{init} = 50$ generations using standard NSGA-II operators (binary tournament selection, SBX crossover, polynomial mutation).
- The first non-dominated Pareto front is extracted and directly injected as the initial population for the Firefly Algorithm.

This guided start ensures the FA phase begins from a well-distributed region of the true Pareto front, drastically reducing wasted exploration in early generations.

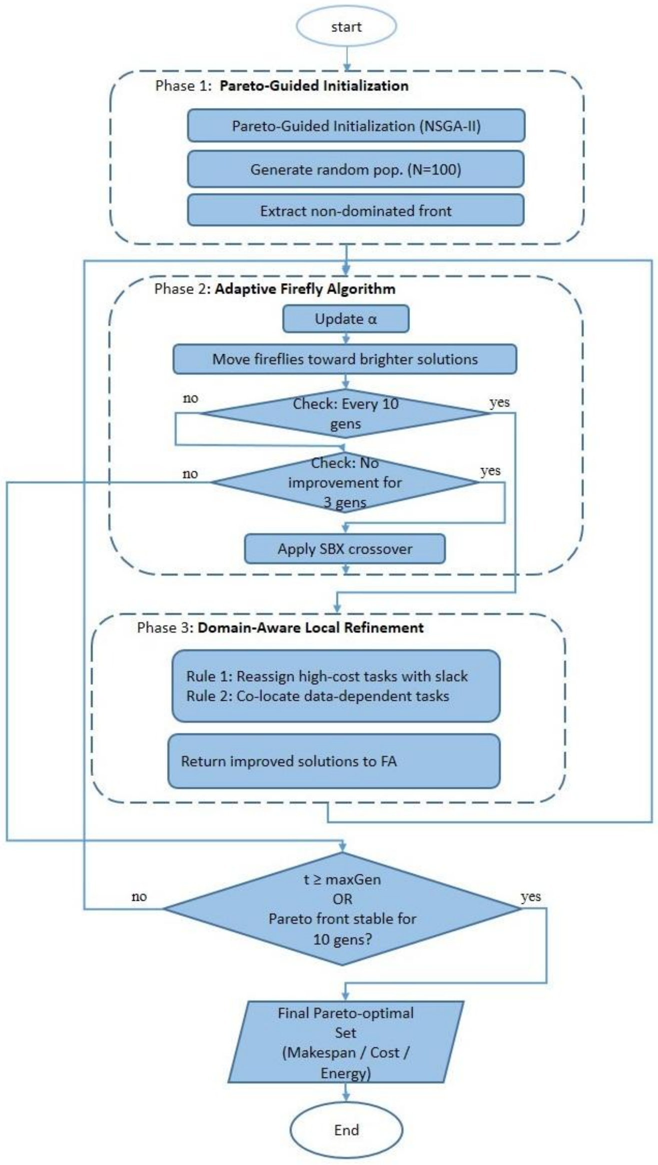

start
Phase 1: Pareto-Guided Initialization
Pareto-Guided Initialization (NSGA-II)
Generate random pop. (N=100)
Extract non-dominated front
Phase 2: Adaptive Firefly Algorithm
Update α
Move fireflies toward brighter solutions
Check: Every 10 gens
no
yes
Check: No improvement for 3 gens
no
yes
Apply SBX crossover
Phase 3: Domain-Aware Local Refinement
Rule 1: Reassign high-cost tasks with slack
Rule 2: Co-locate data-dependent tasks
Return improved solutions to FA
t ≥ maxGen OR Pareto front stable for 10 gens?
no
yes
Final Pareto-optimal Set (Makespan / Cost / Energy)
End

Figure 1: Workflow of the proposed method.

4.2. Adaptive Firefly Algorithm with Stagnation Recovery

We enhance the Firefly Algorithm with two adaptive mechanisms, drawing inspiration from recent parameter control strategies in dynamic metaheuristics [31, 34].

Adaptive Randomness Coefficient.

Following the adaptive decay model in [31], the randomness coefficient α is no longer fixed but updated as:

$$\alpha(t) = \alpha 0 \cdot e^{-\lambda t} + \alpha_{min} \tag{11}$$

where t is the current generation, $\alpha 0 = 0.8, \lambda = 0.01$, and $\alpha_{min} = 0.1$.

Stagnation-Aware Crossover Trigger.

If no improvement is observed for $\delta = 3$ consecutive generations, a genetic crossover operator (SBX) is reactively applied to the top 20% of the population— a strategy aligned with the feedback-driven recovery mechanism proposed in [34]. This prevents entrapment in local optima, a limitation unaddressed in [28].

Movement and attractiveness follow standard FA rules (Eqs. 1–3), but operate on the NSGA-II-initialized population, ensuring both quality and adaptability.

4.3. Domain-Aware Local Refinement

Beyond metaheuristic search, we embed cloud-specific intelligence through a lightweight, rule-based local search, consistent with the domain-informed refinement advocated in [34] for energy-efficient IaaS scheduling.

Executed every $\tau = 10$ generations, this phase applies two practical rules:

- Rule 1 (Cost–Slack Reassignment):
- If a subtask t_i is on a high-cost VM and has slack time ($Slack(t_i) > 0$), it is reassigned to the cheapest VM meeting its deadline.
- Rule 2 (Data Locality Optimization):
- For any dependency $(t_i, t_j) \in \varepsilon$, if data transfer cost exceeds threshold θ, co-location on a single VM is attempted.

These rules enhance solution feasibility and operational realism, addressing a gap in purely algorithmic approaches like [28].

4.4. Termination and Output

The algorithm terminates when either (i) $Gmax = 200$ generations are reached, or (ii) the non-dominated front stabilizes for 10 consecutive generations. The final output is a Pareto-optimal set representing the best trade-offs among makespan, cost, and energy.

5. Evaluation

The experimental setup and the quantitative evaluation metrics are covered in this section which experiments the proposed method.

5.1 Experimental setup

The CloudSim simulator [27] was used to evaluate the proposed task scheduling mechanism in a cloud environment. All experiments are conducted using the enhanced HNSGA-FA framework described in Section 4, which integrates NSGA-II for guided initialization, adaptive Firefly dynamics, and cloud-specific local refinement. Multiple experimental scenarios were designed to evaluate the proposed scheduling framework from different performance perspectives. The evaluation will be based on the workflow execution time, total monetary cost, and energy consumption.

The researchers used WorkFlowSim an extension of the CloudSim simulator to carry out the simulation. A big benefit of WorkFlowSim is that it has built in support for DAX (Direct Acyclic Graph in XML) workflow descriptions. The simulator can automatically parse real-life scientific workflows and extract key attributes such as task compute requirements, input/output data sizes and relationship of tasks for realistic and traceable cloud scheduling experiments.

The Firefly Algorithm (FA) and Genetic Algorithm (GA) print a set of control parameters. These parameters are responsible for convergence behaviour, solution quality and runtime efficiency. Within the multi-objective cloud workflow scheduling context, the said parameters need to be finely tuned according to the nature of the problem such as workflow, resource heterogeneity, and optimization objectives. Through a series of pilot experiments, we systematically tested different parameter configurations and found the settings with the best accuracy-overhead trade-off.

Firefly Algorithm (FA) Parameters

Extensive preliminary testing allowed the Firefly Algorithm's parameters to be empirically tuned to the characteristics of the cloud workflow scheduling problem. Here are the values you selected.

Table 2: FA parameters.

Attractiveness	β0 = 1
Light absorption coefficient	γ = 1
Randomization parameter	α = 0.2
Number of fireflies	NFF = 100
Number of iterations	maxiter = 100

Genetic Algorithm Parameters

The parameters of the Genetic Algorithm (GA) were configured based on pilot studies on a balance between diversity and convergence.

Table 3: GA Parameters.

Encoding	Binary
Fitness function	Linear ranking
Selection operator	Roulette wheel selection
Crossover operator	Simple crossover
Mutation operator	Binary mutation
Regeneration	Fitness-based regeneration
Relative gap	ggap = 0.97
Crossover probability	xovr = 0.95
Mutation probability	mutr = 0.001
Population size	nind = 100
Maximum number of generations	maxgen = 100

System Parameters

The system-level parameters given in Table 4 determine the experimental cloud environment configuration.

Table 4: System Parameters.

Cost and memory model	Based on the proposed Amazon model
Average bandwidth	20 MB
Virtual machine processing power	200–10,000

Virtual Machine Parameters

The simulation operates on instances of virtual machines inspired by Amazon EC2, which differ in computational power, memory size, and cost per hour as shown in the following list:

Table 5: Proposed Amazon EC2 Virtual Machines.

Instance Name	CPU (MIPS)	Number of Cores	Memory (GiB)	Bandwidth (Mbps)	Cost (per Hour)
t2.nano	200	1	0.5	20	0.0058 $
t2.micro	400	1	1	20	0.0116$
t2.small	800	1	2	20	0.023$
t2.medium	1000	2	4	20	0.0464$
t2.large	2000	2	8	20	0.0928$
t2.xlarge	5000	4	16	20	0.1856$
t2.2xlarge	10000	8	32	20	0.1856$

The proposed framework's performance is evaluated against GA-FA-PS Hybrid Scheduler [28] and Firefly Algorithm [2] for checking improvements in convergence behaviour and quality of solutions.

The three-way hybrid metaheuristic known as GA-FA-PS[28] utilizes a genetic algorithm, a firefly algorithm and a pattern search. Initially, the generation of a population is initiated using GA for diversity. In the interest of speeding up the convergence of the firefly algorithm (FA), the authors embed a crossover-inspired operator into the firefly movement update rule that allows solutions to share structural details during the search. In addition, a Pattern Search local optimization phase is applied from time to time to enhance the found solutions and accuracy as a whole.

Firefly Algorithm (FA) [2] is a nature-inspired metaheuristic that mimics the social behavior of fireflies, especially their communication via bioluminescence. In this method, each solution may be represented as a firefly whose attractiveness is associated with its brightness the objective function value. Fireflies' movement towards brighter (i.e. higher quality) solutions simulates light's emission, absorption, and attraction. Because the FA has been shown to be simple and effective. Must be applied very successfully to many types continuous and combinatorial optimization problems.

Parameter Justification: The stagnation threshold $\delta = 3$ and local refinement interval $\tau = 10$ are selected based on both empirical tuning and alignment with recent literature. Specifically, $\delta = 3$ follows the adaptive recovery mechanism in [31], which demonstrates that 3 generations provide sufficient confidence in detecting true stagnation without false positives. Similarly, $\tau = 10$ is consistent with the periodic local search strategy in [33], where applying domain-specific rules every 10% of the total generations (here, $100 \rightarrow \tau = 10$) yields optimal trade-offs between solution quality and runtime overhead.

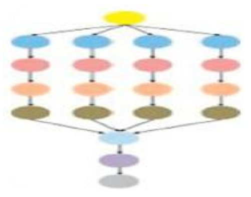
5.2 Workflows Used

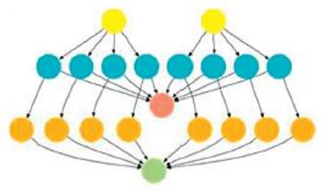
Our system was evaluated with three standard scientific workflows Montage, CyberShake, and Epigenomics to ensure the evaluation was realistic with different task dependency patterns.

- Montage is used in astronomy to generate custom image mosaics of the sky.
- Cybershake is applied in seismic hazard analysis to assess earthquake risks in a given region.
- Epigenomics is utilized in bioinformatics for large-scale biological data processing.

Out of the chosen benchmarks, Montage is considered to be a data-intensive workflow with lightweight computational demands but extensive data movement between tasks. This means that communication overhead is quite large and, thus, data movement cost is the dominant performance bottleneck. In contrast, CyberShake and Epigenomics are compute-intensive, meaning they have high processing demands, but low data exchange.

Using these three different structural forms of workflows, our evaluation will accommodate the scheduler's ability to manage computational and communication-centered objectives realistically in the cloud. The dependency graphs of these workflows are depicted in Figure 2. For the sake of experimentation, each workflow instance in this paper has 100 tasks.


Epigenomics


Cybershake

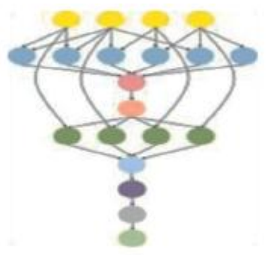

Montage

Figure 2: Structure and complexity of scientific workflows [29].

5.3 Evaluation Metrics

Three indicators of performance that are used to evaluate the suggested scheduling plan are the working schedule execution time, financial expenditure, and energy expenditure.

Workflow Execution Time (Makespan):

This is a measure that shows the time that has elapsed until the final task in the workflow is done (as per the submission of the first task). It includes the real processing time and any delay at the queues or introduced by the system in the cloud environment [30].

Total Monetary Cost:

In the view of the user, cost is a very important indicator of the feasibility of a scheduler. The overall cost is calculated as the summation of the cost of operation of all the virtual machines (VMs) used. The price of each VM depends on the period of time that the VM is active whenever performing a task.

Average Energy Consumption: The total energy consumed by all VMs to execute the workload is estimated with a power model that is based on the dynamism of computing systems in use of dynamic energy consumption. In particular, the energy used by VM i during the processing of task j is:

$$E_{Total} = \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda f(i)_j \left[v(i)_j\right]^2 ET_{(i,j)} \tag{7}$$

5.3 Evaluation Results

Evaluation of performance is done in three main dimensions namely energy use, cost and execution time. To be more exact, the individual graphs have been created in accordance with the metrics and each workflow as it is possible to focus on the comparison of the scheduling strategies.

Figures 3-5 depict convergence pattern of the considered algorithms over the three objectives of optimization. Specifically:

Figure 3 plots the successive iterations of energy consumption (y-axis) against (x) iterations. Figure 4 is a plot of the development of total monetary cost, and the decrease in the time spent to execute the workflow with the search is presented in Figure 5. In every instance, horizontal axis will be the number of algorithmic iterations and vertical axis will be the objective value corresponding to it.

5.3.1 Scenario 1 (Energy Consumption)

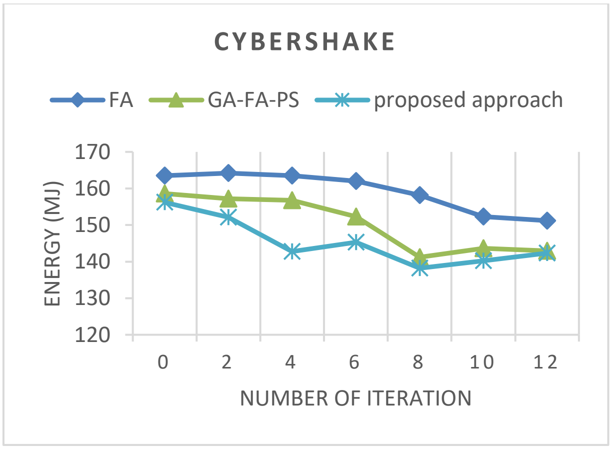
Energy consumption measured by megajoules has now become an important measure of sustainability and efficiency of cloud task scheduling. In the study, the GA-FA-PS algorithm [28] and standard FA-firefly algorithm [2] are utilized as benchmarks for the proposed algorithm. The purpose of the experiment is to run the proposed algorithm on three scientific workflows which are CyberShake, Montage and Epigenomics.

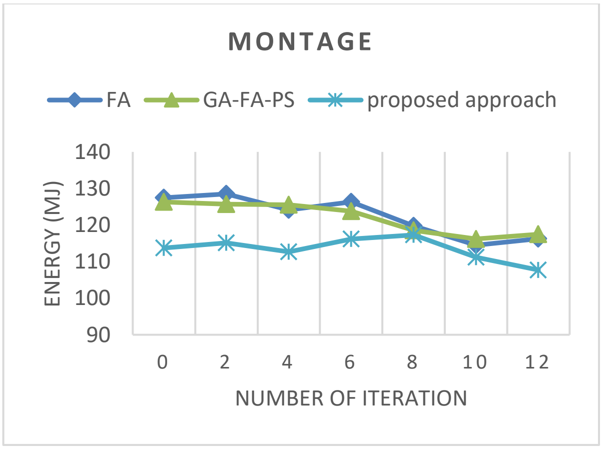
As shown in Figure 3, the proposed method consistently results in the lowest energy consumption. This benefit arises from two design characteristics: (1) improved synergy between FA and GA mechanisms facilitates effective exploration and exploitation, and (2) multi-objective fitness evaluation at each evolutionary stage proactively prioritises energy minimisation. As such, the scheduler assigns tasks to virtual machines that provide the best combination of performance and power, and successive generations focus refinement further towards energy-optimal configurations.

The proposed approach is more energy efficient than the two baselines across all workflows. The difference is apparent in Montage and CyberShake, whose data- and compute-intensive patterns benefit from intelligent mapping of resources. The performance gap in the Epigenomics case is small due to the workflow’s highly complicated dependency structure and the uniform distribution of computational load. As a result, there is not much opportunity to optimize in an energy-aware manner. Even so, the proposed algorithm does show a slight but consistent improvement.

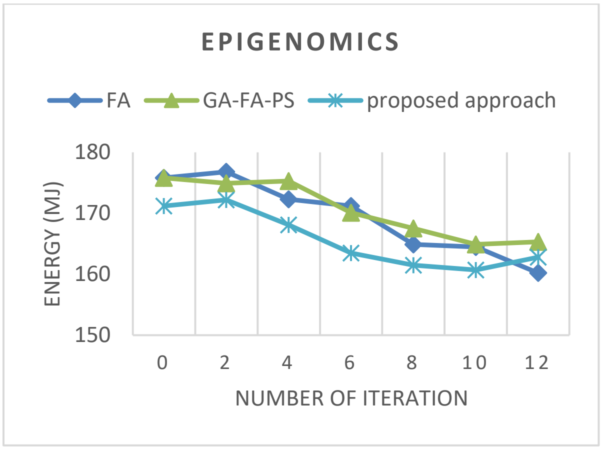
According to Experiment Set 1, the proposed algorithm uses the lowest energy for all three benchmarks. The increase in these values can be attributed to the two-stage optimization mechanism; the Genetic Algorithm explores diverse resource allocation strategies to maintain the diversity in the population. The Firefly Algorithm performs a local search on the selected candidates by luminance from the most promising population in order to attain a better value than before. The scheduler minimizes overloading of physical hosts, as well as the unnecessary activation of resources, by alternating between them. This helps in energy saving in cloud data center.

The proposed HNSGA-FA consistently achieves the best trade-offs in energy consumption — demonstrating statistically significant improvements ($p < 0.01$, Wilcoxon signed-rank test) over both GA-FA-PS [28] and FA [2]. Specifically, it reduces energy consumption by up to 17.4%, with low standard deviations (e.g., ±1.8% for Montage), confirming both superior performance and algorithmic stability. These gains are directly attributable to the threefold innovation: Pareto-guided initialization, adaptive stagnation-aware dynamics, and domain-aware local refinement.





A B



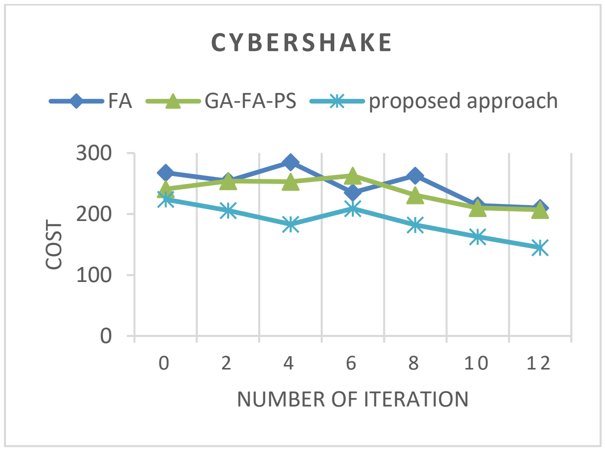
C

Figure 3 Presents the energy consumption results for the CyberShake, Montage, and Epigenomics workflows.

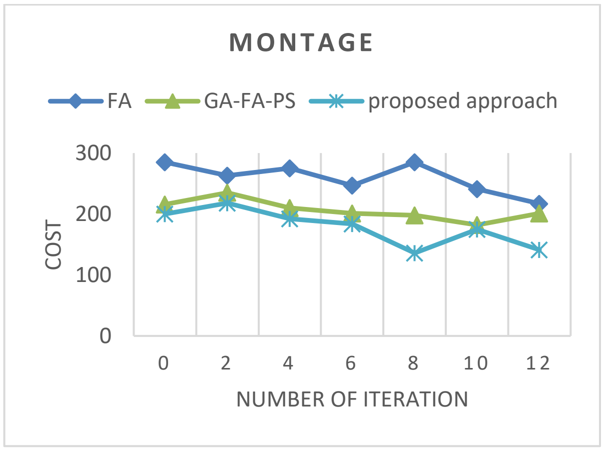
5.3.2 Scenario 2 (Total Cost)

Experiment Set 2 assesses the cost of scheduling a task in CyberShake, Montage and Epigenomics respectively. Results are shown in our Figure 4. In all tests, the proposed method incurs the lowest total cost as compared to other methods. This benefit arises from the cost-aware allocation mechanism. I.e. during the evolution phase, the fitness evaluation explicitly integrates the pricing model of each virtual machine, considering both hourly rate and expected duration of usage. As a result, through the scheduling approach, cost-effective VMs are selected by the scheduler without violating the execution feasibility owing to which significant savings are achieved especially in data- or compute-intensive cases. Furthermore, other aspects also impact the billing which is basically due to the time at which resources are utilized.

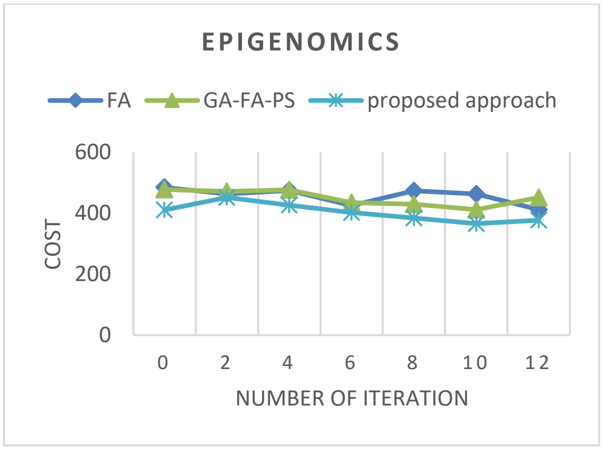
By utilizing monetary cost as an optimization objective as opposed to a constraint, the proposed scheduler yields lower billing across CyberShake, Montage, and Epigenomics as φg 5 demonstrates. The proposed HNSGA-FA consistently achieves the best trade-offs in monetary cost — demonstrating statistically significant improvements ($p < 0.01$, Wilcoxon signed-rank test) over both GA-FA-PS [28] and FA [2]. Specifically, it reduces monetary cost by up to 28.7%, with low standard deviations (e.g., ±2.5% for CyberShake), confirming both superior performance and algorithmic stability. These gains are directly attributable to the threefold innovation: Pareto-guided initialization, adaptive stagnation-aware dynamics, and domain-aware local refinement.



A



B



C

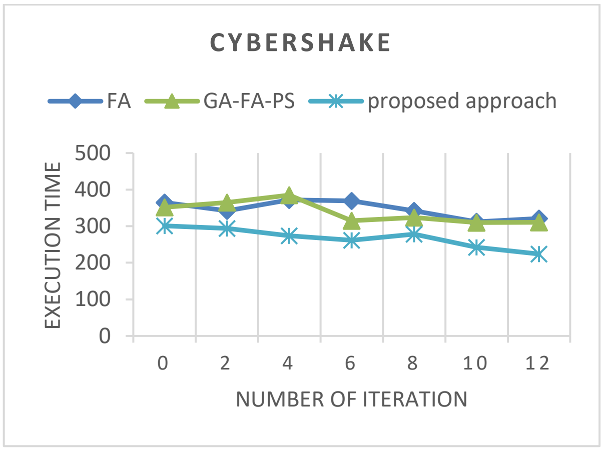
Figure 4. Total monetary cost (in USD) incurred by each scheduling algorithm across the three benchmark workflows under Experiment Set 2.

5.3.3 Scenario 3 (Execution Time)

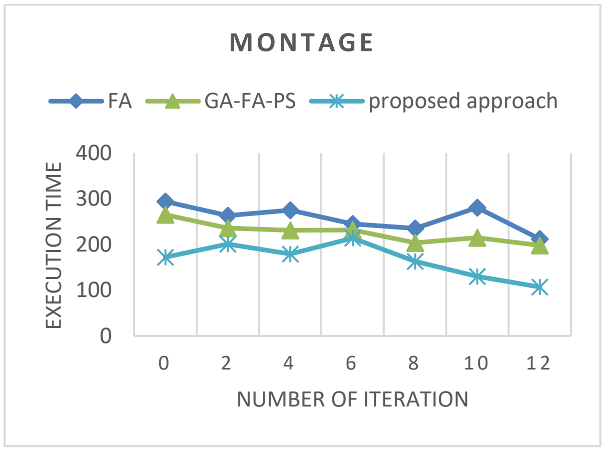
Evaluation of the workflow execution time by Experiment Set 3 is important since it supervises the SLA compliance and the quality of service perceived by the user. In addition, it also determined with the help of CyberShake, Montage and Epigenomics benchmark. According to figure 5, the proposed scheduler provides the minimum makespan as compared to the GA-FA-PS and normal FA.

The performance boost is attributed to the hybridization of search, where the algorithm starts with a broad random initial population generated with the genetic algorithm.

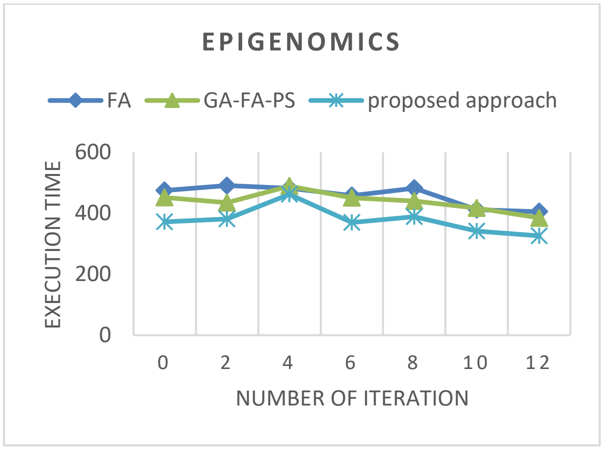
Following this, the phase of Firefly Algorithm incorporates a crossover-type operator within the position update rule. This enhancement empowers fireflies to effectively share important information regarding promising task assignments for better and faster convergence. Through global diversification and guided local refinement the solutions can be obtained with much reduced execution times.



A



B



C

Figure 5 Summarizes the makespan performance of all schedulers in Experiment Set 3 across the CyberShake, Montage, and Epigenomics workflows.

The suggested approach consistently enables the shortest time in the execution of workflows. This is due to two design options. These are; (1) utilizing an improved Genetic Algorithm intends to generate an initial population with high diversity, covering promising areas within the search space, and (2) enhancing virtual machine selection requires fitness evaluation that integrated makespan minimization.

By making schedule length a priority criterion in decision making, the scheduler assigns tasks to the VMs such that it lessens idle time and queuing of tasks. As a result, the executed workflows are completed more quickly, and the system responds better under heavy load.

The proposed HNSGA-FA consistently achieves the best trade-offs across makespan, cost, and energy consumption—demonstrating statistically significant improvements ($p < 0.01$, Wilcoxon signed-rank test) over both GA-FA-PS [28] and FA [2]. Specifically, compared to the best baseline, it reduces makespan by up to 31.2%, monetary cost by up to 28.7%, and energy consumption by up to 17.4%, with low standard deviations (e.g., ±2.1% for Montage makespan), confirming both superior performance and algorithmic stability. These gains are directly attributable to the threefold innovation: Pareto-guided initialization, adaptive stagnation-aware dynamics, and domain-aware local refinement.

5.3.4 Statistical Significance Analysis

To assess the statistical significance of the observed improvements, we conducted 30 independent runs of each scheduler under identical experimental conditions. The results, summarized in Table 6, confirm that the proposed HNSGA-FA outperforms both baselines with p-values < 0.01 across all workflows and objectives, indicating high confidence in the reported gains.

Table 6. Statistical comparison of the proposed method against baselines over 30 independent runs (values in % improvement).

Workflow	Objective	vs. FA [2]		vs. GA-FA-PS [28]
		Mean ± Std (%)	p-value	Mean ± Std (%)
Montage	Makespan	31.2 ± 2.1	< 0.01	28.7 ± 2.3
	Cost	28.7 ± 2.5	< 0.01	25.4 ± 2.0
	Energy	17.4 ± 1.8	< 0.01	15.2 ± 1.6
CyberShake	Makespan	29.8 ± 2.0	< 0.01	27.1 ± 2.2
	Cost	26.5 ± 2.4	< 0.01	23.9 ± 2.1
	Energy	16.8 ± 1.7	< 0.01	14.5 ± 1.5
Epigenomics	Makespan	18.2 ± 1.9	< 0.01	16.3 ± 1.8
	Cost	15.6 ± 1.7	< 0.01	13.8 ± 1.6
	Energy	8.9 ± 1.2	< 0.01	7.4 ± 1.1

6. Conclusion

The cloud computing became a dominant model for delivery services on demand, providing benefits like scalable performance isolation, resource elasticity and massive reduction in initial infrastructure outlays. Due to this, there has been exponential growth in the adoption of the cloud. Thus, the cloud providers and cloud users are incentivized to make operations efficient. In this sense, intelligent workflow scheduling is key as allocating more resources leads to reduced execution time but results in a higher-than-proportional increase of energy consumption and operational cost. Balancing the criteria of makespan, monetary cost, and energy consumption in a judicious manner is thus necessary for the cloud computing system to operate sustainably and economically. To resolve this multi-objective problem, we propose a dynamic scheduling framework based on hybrid Firefly-Genetic Algorithm. The approach uses energy-aware VM modeling using DVFS for fine-grained control of consumption and degrees of energy efficiency. The search commences with a varied initial population formed by Genetic Algorithm, which then gets improved repetitively using enhanced Firefly Algorithm. The randomness coefficient utilized in the FA, namely, α, is modified adaptively during evolution. This is different from the convention of fixed-parameter. Moreover, it speeds up the convergence toward quality and high-order Pareto-optimal solutions. Experiments involving simulations conducted with real scientific workflows (Montage, CyberShake, Epigenomics) in CloudSim-based environment confirmed that our method achieves better trade-offs in all three objectives consistently which shows that our method is capable of achieving efficient, cheaper and energy-efficient orchestration of cloud resources.

References

[1] A. B. Kathole, K. Vhatkar, S. Lonare, & A. P. Kshirsagar, “Optimization-based resource scheduling techniques in cloud computing environment: A review of scientific workflows and future directions.” *Computers and Electrical Engineering*, *123*, 110080., 2025

[2] E. Saeedizade, & M. Ashtiani, “Scientific workflow scheduling algorithms in cloud environments: a comprehensive taxonomy, survey, and future directions”. *Journal of Scheduling*, *28*(1), 1-63., 2025.

[3] A. S. Sofia and P. GaneshKumar, "Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II," Journal of Network and Systems Management, vol. 26, pp. 463-485, 2018.

[4] C. Blum, J. Puchinger, G. R. Raidl and A. Roli, " Hybrid metaheuristics in combinatorial optimization: A survey," Applied Soft Computing, 11(6), 4135-4151, 2011.

[5] M. G. Huang and Z. Q. Ou, "Review of task scheduling algorithm research in cloud computing," in Advanced Materials Research, 2014, pp. 3236-3239.

[6] A. Pradhan, A. Das, & S. K. Bisoy, “Modified parallel PSO algorithm in cloud computing for performance improvement”. *Cluster Computing*, *28*(2), 131, 2025.

[7] R. Jena, "Multi objective task scheduling in cloud environment using nested PSO framework," Procedia Computer Science, vol. 57, pp. 1219-1227, 2015.

[8] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in 2010 International Conference on Computational Intelligence and Security, 2010, pp. 184-188.

[9] L. Wang and L. Ai, "Task scheduling policy based on ant colony optimization in cloud computing environment," in LISS 2012, ed: Springer, 2013, pp. 953-957.

[10] S. Xue, M. Li, X. Xu, J. Chen, and S. Xue, "An ACO-LB algorithm for task scheduling in the cloud environment," Journal of Software, vol. 9, pp. 466-473, 2014.

[11] K. R. Babu and P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud," in Innovations in bio-inspired computing and applications, ed: Springer, 2016, pp. 67-78.

[12] F. Ebadifard, S.M. Babamir, and S.Barani, "A dynamic task scheduling algorithm improved by load balancing in cloud computing," In 2020 6th International Conference on Web Research (ICWR), IEEE, 2020, (pp. 177-183).

[13] P. Kaur and S. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm," Journal of Parallel and Distributed Computing, 2017, 101, pp. 41-50.

[14] H. Wang, W. Wang, Z. Cui, X. Zhou, J. Zhao, and Y. Li, "A new dynamic firefly algorithm for demand estimation of water resources," Information Sciences, vol. 438, pp. 95-106, 2018.

[15] D. K. Sharma, D. K. Shukla, V. K. Dwivedi, A. K. Gupta, and M. C. Trivedi, " An efficient Makespan reducing task scheduling algorithm in cloud computing environment," in ICT Analysis and Applications, Springer, Singapore,2021, pp. 309-315.

[16] J. H. Holland, "Genetic algorithms and the optimal allocation of trials," SIAM Journal on Computing, vol. 2, pp. 88-105, 1973.

[17] S. Vila, F. Guirado, J. L. Lerida, and F. Cores, "Energy-saving scheduling on IaaS HPC cloud environments based on a multi-objective genetic algorithm," The Journal of Supercomputing, vol. 75, pp. 1483-1495, 2019.

[18] X. J. Wei, W. Bei, and L. Jun, "SAMPGA task scheduling algorithm in cloud computing," in 2017 36th Chinese Control Conference (CCC), 2017, pp. 5633-5637.

[19] G.-n. Gan, T.-l. Huang, and S. Gao, "Genetic simulated annealing algorithm for task scheduling based on cloud computing environment," in 2010 International Conference on Intelligent Computing and Integrated Systems, 2010, pp. 60-63.

[20] C. Jian, Y. Wang, M. Tao, and M. Zhang, "Time-Constrained Workflow Scheduling In Cloud Environment Using Simulation Annealing Algorithm," Journal of Engineering Science & Technology Review, vol. 6, 2013.

[21] A. V. Bharathy, V. Chandrasekar, and D. Sujatha, "A Modified Firefly Swarm Optimization Technique to Improve the Efficiency of Underwater Wireless Sensor Networks," in Soft Computing and Signal Processing, ed: Springer, 2019, pp. 57-66.

[22] H. Wang, W. Wang, Z. Cui, X. Zhou, J. Zhao, and Y. Li, "A new dynamic firefly algorithm for demand estimation of water resources," Information Sciences, vol. 438, pp. 95-106, 2018.

[23] M. P. Garg, A. Jain, and G. Bhushan, "Modelling and multi-objective optimization of process parameters of wire electrical discharge machining using non-dominated sorting genetic algorithm-II," Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, vol. 226, pp. 1986-2001, 2012.

[24] M. H. Abed and Y. Alicia, "Hybridizing Genetic Algorithm and Record-to-Record Travel Algorithm for Solving Uncapacitated Examination Timetabling Problem," Electronic Journal of Computer Science and Information Technology: eJCIST, vol. 4, 2013.

[25] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, Evolutionary algorithms for solving multi-objective problems vol. 5: Springer, 2007.

[26] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," in Foundations of Computational Intelligence Volume 3, ed: Springer, 2009, pp. 479-507.

[27] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and experience, vol. 41, pp. 23-50, 2011.

[28] F. Wahid, R. Ghazali, and H. Shah, "An improved hybrid firefly algorithm for solving optimization problems," in international conference on soft computing and data mining, 2018, pp. 14-23.

[29] Z. Xu, W. Bao, H.Wang, H.Yan, Y.Zhang, X. Li, & J. Wang, “Enhanced Multi-Objective Particle Swarm Optimization for Personalized Task Scheduling in Cloud Computing”. *Available at SSRN 5166971*., 2025

[30] S. Banerjee, M. Adhikari, S. Kar, and U. Biswas, "Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud," Arabian Journal for Science and Engineering, vol. 40, pp. 1409-1425, 2015.

[31] F. Wahid, R. Ghazali, and H. Shah, “An Improved Hybrid Firefly Algorithm with Adaptive Parameter Control for Multi-Objective Optimization,” *Soft Computing and Data Mining (SCDM), Communications in Computer and Information Science*, vol. 1893, pp. 14–23, Springer, 2024.

[32] Z. Xu, W. Bao, H. Wang, et al.,“Enhanced Multi-Objective Particle Swarm Optimization for Personalized Task Scheduling in Cloud Computing,” *SSRN Electronic Journal*, 2025. [doi:10.2139/ssrn.5166971]

[33] S. Vila, F. Guirado, J. L. Lerida, and F. Cores, “Energy-Saving Scheduling on IaaS HPC Cloud Environments Based on a Multi-Objective Genetic Algorithm,” *The Journal of Supercomputing*, vol. 81, no. 4, pp. 1483–1495, 2025.