Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Energy Optimized Workflow Scheduling in IaaS Cloud: A Flower Pollination based Approach†

Sahani Pooja Jaiprakash*[1,2] | Harsh Kumar Arya[1] | Indrajeet Gupta[1] | Tapas Badal[1]

[1]School of Computer Science Engineering and Technology , Bennett Univarsity, Greater Noida in Uttar Pradesh, India

[2]Department of CSE, Institute of Technical Education Research (ITER), Siksha 'O' Anusandhan (Deemed to be University), Odisha , India

Correspondence
*Sahani Pooja Jaiprakash, School of Computer Science Engineering and Technology, Bennett Univarsity, Greater Noida in Uttar Pradesh, India . Email: sahanipooja26@gmail.com

Present Address
Department of CSE, Institute Of Technical Education Research (ITER), Siksha 'O' Anusandhan (Deemed to be University), Odisha, India

Summary

The energy consumption of cloud data centers is a critical concern that could affect both the environment and the availability of energy resources. For this, the global community and industries are taking measures to address this issue that is caused by the high electricity consumption of servers, Heating, Ventilation, and Air Conditioning (HVAC) in the data centers. With this context, this paper presents a novel approach for scheduling energy-efficient workflows (EEWS) in cloud computing using the MaxUtil model. The proposed approach incorporates the flower pollination algorithm (FPA), a popular meta-heuristic algorithm inspired by nature. The primary objectives of the proposed scheduling scheme are to minimize energy consumption and workflow processing time (makespan). The proposed algorithm involves two key phases: (i) assigning tasks to available virtual machines (VMs) and (ii) scheduling the tasks based on optimal criteria. As per our knowledge, this is the first study that focuses on optimizing energy consumption and makespan in cloud computing workflow scheduling using FPA. The proposed approch employs an effective representation of pollen and dynamic fitness function with multi-objective. The advantage of FPA lies in its speed of convergence and providing feasible solutions. Extensive studies have been conducted across five different scientific workflows from various fields. The proposed algorithm outperforms traditional workflow scheduling algorithms based on particle swarm optimization (PSO), gravitational search algorithms (GSA) and genetic algorithm (GA). The proposed algorithm outperforms GA, PSO, and GSA in the majority of cases, according to simulation findings. In addition, a well-known statistical test known as variance analysis (ANOVA) is used to validate the experimental results of the suggested algorithm. Based on the result's of ANOVA test, the article claims that the suggested algorithm is superior to existing methods.

KEYWORDS:
Workflow Scheduling, Virtual Machine, Energy Efficient, Makespan, ANOVA, Flower-Pollination

1 | INTRODUCTION

Due to rapid shifting of on-premise IT infrastructure to cloud computing, the energy consumption of data centres has become a crucial issue. A data centre is a place where the actual data are processed and send back to cloud users. While processing data,

†A Flower Pollination based Energy Optimized Workflow Scheduling.

workflows and applications, the data centres consume high amount of energy and this energy usage has became a serious concern for environmental imbalance and increasing the operational cost. In cloud computing, workflow scheduling is a practice of allocating tasks to the cloud resources in order to fulfill the user requirements as per the service level agreement (SLA).[1,2,3,4,5]. It has many business and scientific uses, such as in astronomy, weather predicting, medicine, and bio-informatics. Most of the time, these tasks are big because they have a lot of independent and/or interdependent processes that need a lot of computing, memory, communication, and storage power[6]. So, workflow/task scheduling can be defined as the process of assigning computational jobs to virtual machines (VMs) in a way that makes the best use of resources. This scheduling is identified as a NP complete problem.

McKinsey reported[7] in 2010 that data centres expended approximately 11.5 billion dollars on energy which doubles the energy expenditures in every five years. Furthermore, it can be observed that the worldwide data centres have consumed an estimated range of 220-320 terawatt-hours (TWh) of electricity in year 2021, which is equivalent to approximately 0.9-1.3% of the total electricity demand of the entire globe[8]. So, there is a need of sustainable options which can reduce the energy consumption, like green data centers. As we know, environmental sustainability is a prime focus of technological advancement and modern way of using cloud computing, the global research community and cloud industries have been developing eco-friendly research outcome that are satisfying the criteria of green computing[9].

Due to the distributed nature of cloud resources, cloud servers face power-related challenges to handle massive workloads and managing cloud resources while processing the tasks. So, here is a need of intelligent task scheduling and resource provisining approach on the backend of underlying cloud infrastructure. In this regards, various researchers are being publishing their researches that aim to schedule user application in minimum processing time while minimizing energy consumption. The existing research works are categoried in three broad categories that are heuristic, metaheuristic and hybrid algorithms[10].

Although using virtualization technique the CSP offers a wide range of cloud resources in the form of virtual servers. These virtual servers can work stand alone or in the from of clusters. The clusters are logical VM pool, those are handled by virtual machine manager (VMM) or hypervisor. The VMM is responsible for creating the logical VM pool or VM cluster from the physical machines or servers[11,12]. Here, several factors are in considration while developing the energy efficent task scheduling algorithms. Such as managing the wide range of hetrogeneous VMs, dealing with unpredictable workloads, user specified SLA, operational excellence and performance efficiency[13,14]. For this, the CSP targets on minimizing energy consumption and maximazing cloud resources utilization.

In heuristic-based approach, First Come First Serve (FCFS), Shortest Job First (SJF), and Round Robin (RR) are widely used algorithms. Lets assume, if there are n tasks and m available VMs, then the possible ways to map the tasks to the available VMs can be m^n. However, this approach for finding an optimal solution can be computationally expensive, particularly for large values of n and m. As a result, heuristic-based approaches are preferable for finding a quick solution instead of the best possible one[15,16,17]. To overcome this limitation, metaheuristic algorithms, such as genetic algorithm, ant colony optimization, particle swarm optimization, and so on are already used in task scheduling for searching the efficient pattern in the solution space. Similarly, hybrid algorithms use heuristic and metaheuristic methods, among others, to improve performance of task scheduling. Although each algorithm has its own strengths and weaknesses, metaheuristic approaches are generally more effective than heuristic-based techniques[18]. Recent studies in cloud computing have demonstrated that combining metaheuristic approaches can produce better results by addressing multiple objectives simultaneously[19].

In this research paper, we used MaxUtil model and proposed an energy-efficient workflow scheduling (EEWS) algorithm that aims to minimize both energy consumption and makespan. Even previous studies have considered these objectives but they have often overlooked an important aspect of scheduling, which is CPU performance variability[20]. Specially, CPU performance can vary by up to 24%[21], which can have a significant impact on scheduling. Furthermore, other real-world factors, such as VM boot and shut-down times, have also been neglected in previous research. In our proposed approach, we have taken all of these factors for considertation. Our proposed method is a meta-heuristic algorithm inspired by pollination of flowers, to develop our scheduling system. Proposed algorithm is a multi-objective dynamic function with primary focus on minimize the energy consumption and makespan, while scheduling the workflow. This method considers the trade-off between performance and power consumption and attempts to find the optimal point where energy consumption is minimized while meeting the performance requirements. The below is the significant contributions of this article:

- We present a systematic method for addressing the workflow scheduling problem in a heterogeneous cloud environment.
- An efficient modified dymanic fitness function of the flower pollination approach is also proposed.

- We design a novel energy-efficient meta-heuristic algorithm related to the flower pollination approach to improve multiple network parameters such as makespan and energy consumption.
- The suggested algorithm is thoroughly simulated using five benchmark scientific workflow applications, and its simulation results are compared with three baseline algorithms to demonstrate its superiority over the existing ones.
- A statistical analysis is done to check the validity and reliability of the results.

The remainder of the paper is divided into the following sections. Task scheduling in cloud computing is examined in section 2. The cloud model and the problem formulation are described in section 3. The proposed algorithm is described and illustratrated in section 4, then the simulation results are presented in section 5. Finally the paper is concluded in section 6.

2 | RELATED WORK

For the sake of societal welfare, any computing paradigm that strikes a healthy balance between its effects on the economy and the environment is always preferable [12]. There have been many different attempts made to boost the functionality of cloud data-centres while also increasing their energy efficiency as well as to reduce the CO_2 emissions and energy costs associated with these facilities [16]. The impact of energy consumption always implies professionalism for developing the cloud computing's energy-efficient task scheduling [22,23,24]. To this end, numerous heuristic and meta-heuristic scheduling approaches have been presented. In this part, we briefly examine previous studies that have addressed the issue of scheduling cloud-based workflows.

In [15], two scheduling strategies, ECTC and MaxUtil were described that are based on energy-aware task consolidation. Unfortunately, both the heuristics do not include the idle power use of accessible virtual resources. Moreover, both the algorithms have simulated only for the independent task not for workflow applications. In the same subsequent, a maximum effective reduction (MER) algorithm also have been proposed by Lee et al. [25]. This work deal with the trade-off between increased makespan and efficient usage of resources. For this, the algorithm required initial schedule plan generation of the given workflow as per the resource availability after that the MER algorithm works. Although, the design of energy efficient scheduling algorithm should not be only goal, it should be accompanied with other objectives also like reduction of workflow processing time (makespan) and meeting of the Quality of service (QoS) parameters.

Vasile et al. [26] suggested a hierarchical clustering-based scheduling algorithm which is the hybridization of task clustering and resource clustering in to different groups. Therefore, this work is titled a hybrid scheduling algorithm for cloud environments based on resource aggregation ($HySARC^2$). Their work accommodated by various different scheduling approaches for independent tasks and DAG applications. Hsu et al. [27] offered yet another impressive study for energy optimization in the cloud computing environment. In their work, an energy-aware task consolidation (ETC) scheme has been proposed which is far better to reduce the energy utilization against the MaxUtil algorithm [15]. The experimental results of ETC claim upto 17% considerably improvement reduce power consumption over the MaxUtil.

Kim et al. [28] suggested a new technique for VM scheduling which relies on energy budget. Even though, in this approach, the error rate of the expected energy consumption was below 5% of the total usage of energy. Srikantaiah et al. [17], have proposed a new solution for task scheduling problem. The solution is founded by using bin packing problem and consolidation of tasks. Wen et al. [29] presented a hierarchical scheduling approach for the user application to reduce power usage. This scheme's main drawback was that the application preferred lower-temperature resource nodes during task-resource mapping. Mohammed et al. [30] suggested an energy-efficient Flower Pollination Methodology based on Dynamic Switching Probability for VM allocation. The framework works quickly to locate a solution that is close to optimal and strikes a balance between local and global search. E-FPA outperforms FFD, OEM, and GAPA by 24.9%, 21.5%, and 21.8%, respectively.

Tran et al. [31] introduces the Q-learning approach in order to address the issue of directed acyclic graph tasks in data-centres. To reduce energy usage, k-means clustering and dynamic VM migration have been used to create energy-efficient dynamic resource management. Service quality and lifespan are prioritised to reduce DC energy usage and resource under utilization. Mohammed et al. [18] developed a top-notch Multi-Objective Hybrid Floral Pollination Resource Consolidation (MOH-FPRC) strategy. Global searches use flower pollination, while local results use the Dynamic Local Neighborhood Search algorithm. Dynamic clustering and a strong migration mechanism lower Service Level Agreement (SLA) violations and energy usage. A VM allocation technique based on the Interference Attentive Genetic Algorithm (IAGA) has been suggested by Bloch et al. [32] to lessen SLAV and performance deterioration on the IaaS platform.

Elaziz et al.[33] proposed MSDE, a meta-heuristic algorithm based on MSA and DE, to improve cloud task scheduling. They focused on reducing makespan. CloudSim simulations were compared to SJF, RR, PSO, WOA, and MSA algorithms. MDSE had a faster makespan and throughput than other methods. The method was time-intensive. Using an unique GWO swarm intelligence-based strategy, the authors[34] developed the solution to the job scheduling issue in a cloud setting. They had the only purpose of reducing the makespan time. They carried out tests in the CloudSim simulation, and the outcomes are contrasted with those obtained using the following known algorithms: FCFS, min-max, ACO, and PBACO. By attaining a smaller makespan value, the proposed approach performed better than existing algorithms. In[7], a workflow scheduling problem that involves both reserved and on-demand instances is taken into consideration. This makes it a more practically applicable solution. The authors constructed a mathematical model consisting of inflexible and flexible tasks with the help of the two different resource renting strategies. An innovative multiple sequence-based technique for determining the earliest possible end time is presented, and then compared to other state-of-the-art methods of doing the same thing.

In[35], the authors developed a novel meta-heuristic algorithm called Improved WOA for Cloud task scheduling (IWC) based on WOA to optimize the solution for task scheduling problem in a cloud environment. They employed a multi-objective of minimizing the monetary and computational cost. The proposed algorithm giving lower computational cost, higher resource utilization, and lower monetary cost. However, it had high scheduling overhead and it didn't consider workflow scheduling. Shukri et al.[36] proposed the Enhanced Multi-Verse Optimizer (EMVO) algorithm to enhance cloud task scheduling. They focused on reducing makespan. They tested MVO and PSO algorithms in a simulated setting. The algorithm had the lowest makespan, maximum throughput, and highest resource utilisation. Velliangiri et al.[37] used a novel meta-heuristic algorithm known as Modified Electro Search (HESGA) to improve cloud task scheduling. The authors executed CloudSim simulations with a multi-objective of minimising makespan and improving cloud service provider resource utilisation. The CPU's optimum energy efficiency ratio interval and the need for prompt processing are used to schedule tasks in[38]. Particle swarm optimization is one way to solve this multi-objective optimization method. With this strategy, data processing for the local controller in the forwarding path is scheduled.

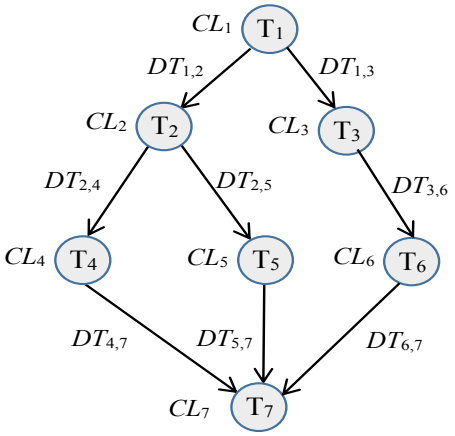
In[39], the authors introduced HHOSA, a meta-heuristic algorithm based on HHO and SA, to refine cloud task scheduling problems. They focused on reducing maketime. For large-scale scheduling issues, it converged faster as the search space expanded. According to the lituralure survey, authors have addressed a particular purpose in several prior studies. Few of them have addressed the issue of makespan and energy use separately[15,26,33,35,37,30]. The existing approaches didn't consider the practical scheduling constraints, such as the boot and shut-down time of VMs and processor performance variability[20]. In this contrast, our proposed method, EEWS, addresses these deficiencies and aims to minimize both energy consumption and makespan during workflow scheduling.

3 | MODELS AND TERMINOLOGIES

In this section, workflow model defines the tasks that need to be executed in a specific order, a cloud deployment model consists the different pools of VMs, then an energy model that is based on Max_Util[27] is a mathematical representation of the energy consumption of a cloud system. This section also consistent of notations and preliminaries.

3.1 | Workflow Model

A workflow describes the sequence of tasks that need to be executed in a specific order to achieve a specific goal, such as data processing or web service invocation. The flow of data or instructions through the multiple jobs from initiation to completion is termed as a workflow. This whole process provides a complete sense of relevant information in the form of the workflow which can be defined by a directed acyclic graph (DAG). A workflow W_f is defined as $W_f = (T, E)$ where, $T = \{T_1, T_2, \cdots, T_n\}$ is set of tasks and E is the set of amount of data transfer from task T_i to task T_j i.e., $T_i \rightarrow T_j$. In the given workflow W, each task node contains the computational load (CL) in the form of million instruction (MI), and each connecting edge E is the set of amount of data transfer (DT) in Megabytes (MBs) from task T_i to task T_j. A task of any workflow W_f cannot be started until its all parent tasks are fully executed. If there is no parent task of any task node in a workflow then it is called an entry task. Similarly, if there is no child task of any task node then it is known as an exit task node. In the case of multiple entry tasks and multiple existing tasks, the presence of pseudo entry task and exit task are to be assumed respectively. The DAG representation of a workflow is shown in Figure 1.



$DT_{i,j}$: Data Transfer (in MB) from task T_i to task T_j

CL_i : Computational Load (in MI) of task T_i

Figure 1 A sample DAG representation of a workflow

Table 1 Notations and Definitions

Notations	Definitions
W_f	A workflow application
n	In a given workflow (W_f), the number of task nodes
T_i	i^{th} task of the given workflow (W_f)
$pollen\_size$	Number of the pollens
$E_c^{VM_j}$	Consumed energy by the CPU of (W_f)
E_0	Consumed energy of VM_j in idle state
$Mksn$	Makespan
D	Variability in processor speed, rated 0-1
$ET_{T_j}^{VM_k}$	Execution time of task T_j on VM_k
$spdV_j$	Decrease in processor speed on a scale of 0 to 1 when T_i is operating.
dV_j	Reduction of the processor's input voltage while T_i is operating.

3.2 | Cloud Model

A cloud model is a set of characteristics that define the different types of cloud computing services and deployment models. In Figure 2, the cloud model is taken as an IaaS cloud service model and it provides VMs pools, which are responsible for task execution. The created m number of VMs are represented by $VM = \{VM_1, VM_2, VM_3, \ldots, VM_m\}$. A VM may belong to either same physical machine or different physical servers. These physical servers are the core infrastructure on which the VMs are deployed. Here, a virtual machine manager (VMM) is introduced for the VM management (i.e., VM creation, VM deletion, & VM suspension). The VMM creates the logical VM pool (VM cluster) among the participating VMs of physical servers.

3.3 | Basic Preliminaries

(a) **Task Runtime or CPU Time** ($Runtime(T_i)$): It is the processing time of any task T_i is defined as dividing its computational load CL from the MIPS speed of virtual machine VM_j. It is represented as follows:

$$Runtime(T_i) = \frac{Computational\ Load\ of\ T_i\ (in\ MI)}{Speed\ of\ VM_j\ (in\ MIPS)} \tag{1}$$

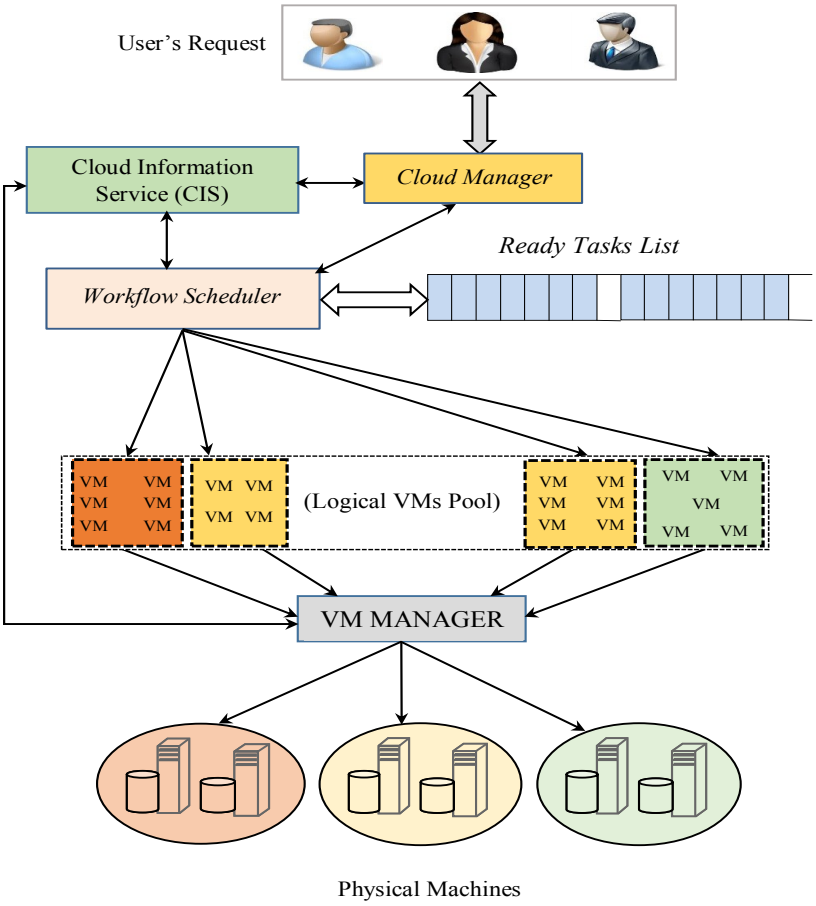


Figure 2 Cloud model for workflow processing

(b) **IO Time** ($IO\_Time(T_i)$): It is the time for transferring the all data files from the predecessor tasks set T_p to the current task T_i using the IO speed in MBPS (IO_{Speed}). It is expressed as follows:

$$IO\_Time(T_i) = \frac{F_{Total}\ in\ MB}{IO_{Speed}\ in\ MBPS} \tag{2}$$

where F_{Total} is the summation of all the data transfer (DT) files at task T_i from all the predecessor task set T_p. It can be explained as follows:

$$F_{Total} = \sum_{p=1}^{|p|} DT_{p,i} \tag{3}$$

$|p|$ is the total number of predecessor tasks of T_i.

(c) **Execution Time of Task** (T_i) ($Exe\_Time(T_i)$): It is the actual required execution time of task T_i which is defined as the summation of $Runtime(T_i)$ and $IO\_Time(T_i)$.

$$Exe\_Time(T_i) = Runtime(T_i) + IO\_Time(T_i) \tag{4}$$

(d) **CPU Utilization** (CPU_Util): During the processing of the task T_i on the VM_j, the utilization of CPU of VM_j is defined as the ratio of $Runtime(T_i)$ and $Exe\_Time(T_i)$. It is expressed by the following formulation:

$$CPU\_Util(VM_j) = \frac{Runtime(T_i)\ or\ CPU\ Time(T_i)}{Exe\_Time(T_i)} \tag{5}$$

Table 2 Notations and Definitions

Notation	Definition
N	Population size.
v_j	j^{th} VM.
m	Number of available VM.
$\mathbb{P}_i$	It represents the i^{th} pollen.
$\mathbb{G}_{best}$	It's the best pollen we've come across upto now.
P_switch	FPA switching probability.
$\mathbb{B}_{pollen}$	Initiale best pollen
α	It calculates fitness by weighting makespan and energy.
tk_i	i^{th} task in pollen.
fit_i	The i^{th} pollen's fitness value dependent on its makspan and energy.
β	Energy makespan equity component. The application's importance and urgency decide its SLA value.
deg_{v_j}	VM v_j performance degradation.
σ	A random element in the pricing model.
δ	Substitution threshold mass.
γ	Minor constant that controls gravity constant declination.

In this paper, we have considered some constraints and assumptions as addressed in[20]. The notations which have been considered throughout the paper are given in Table 2 and below are few useful definitions.

Definition 1. To compute effective *CPU cycles* for task execution, we take performance variation for the VMs into account. The causes for this diversity are related to the heterogeneity and common nature of the underlying hardware's infrastructure. According to a survey[21] , the total performance variability of Amazon's EC2 cloud is 24%. Performance variance is given as deg_{v_j} for the VM v_j. Thus, the execution time of task tk_i on VMv_j may be represented using deg_{v_j} as:

$$ET_{tk_i}^{v_j} = \frac{Load(tk_i)}{Capacity(v_j) \times (1 - deg_{v_j})} \tag{6}$$

where $ET_{tk_i}^{v_j}$ is the time at which task tk_i was executed on VM v_j, Load (tk_i) is the task's computing load, and Capacity (v_j) is VM v_j's computational capacity.

Definition 2. When determining the expense of the execution, the unit chargeable time τ is taken into consideration. Even if we use the leased VM for a period of time that is shorter than τ, we will still be charged for the entire time period.

Definition 3. Whenever a VM is acquired, an initial boot time is required. Therefore, we incorporate VM start time into the makespan calculation. In order to release the provisioned VM , we also evaluate the VM shutdown duration.

Definition 4. The assumption is that each VM has a nearly equal bandwidth connection.

3.4 | Energy Model:

This paper uses MAX-Util energy model[27,40,21,5], new which is based on a consolidation of tasks and CPU utilization of the VM. The consumed energy (E_c) is estimated by various levels of CPU utilisation and higher utilization of CPU always demands higher energy consumption. Hence, consumed energy (E_c) is calculated as a function of CPU utilisation and energy consumption rate. The bandwidth of logical resource clusters in the cloud model is variable and due it, the power consumption by the VMs also varies. Therefore, power consumption is solely affected by CPU and available bandwidth usages.

$$E_c^{VM_j} = \begin{cases} \gamma\ W/s \times (idle\_Time_{T_\phi}^{VM_j}), & if\ VM_j\ is\ idle, & level\ 0 \\ (\delta+\gamma)\ W/s \times Exe\_Time_{T_i}^{VM_j}, & if\ 0\% < CPU\_Util(VM_j) \leq 20\%, & level\ 1 \\ (3\delta+\gamma)\ W/s \times Exe\_Time_{T_i}^{VM_j}, & if\ 20\% < CPU\_Util(VM_j) \leq 50\%, & level\ 2 \\ (5\delta+\gamma)\ W/s \times Exe\_Time_{T_i}^{VM_j}, & if\ 50\% < CPU\_Util(VM_j) \leq 70\%, & level\ 3 \\ (8\delta+\gamma)\ W/s \times Exe\_Time_{T_i}^{VM_j}, & if\ 70\% < CPU\_Util(VM_j) \leq 80\%, & level\ 4 \\ (11\delta+\gamma)\ W/s \times Exe\_Time_{T_i}^{VM_j}, & if\ 80\% < CPU\_Util(VM_j) \leq 90\%, & level\ 5 \\ (12\delta+\gamma)\ W/s \times Exe\_Time_{T_i}^{VM_j}, & if\ 90\% < CPU\_Util(VM_j) \leq 100\%, & level\ 6 \end{cases} \tag{7}$$

In literatures [15,25,27,40,21,5], it is found that the CPU runs in two states: (i) idle state and (ii) running state. In [27,40], it is concluded that energy consumption and CPU utilization have not the linear relationship. Eq. 7 clearly formulates the entire process of energy consumption at all levels with respect to CPU utilisation of VM_j. There are seven different levels of CPU utilisation, ranging from idle to various CPU running states. To make it more realistic, we include both CPU and IO time in this formulation. In Eq. 7, $\gamma\ W/s$ (Watts per second) is the rate of energy consumption when the VM_j is idle, and W/s is an additional energy consumption when $0\% < CPU\_Util(VM_j) \leq 20\%$. Similarly, the various combination of $\gamma\ W/s$ and $\delta\ W/s$ are well explained in the Eq. 7 for the remaining CPU utilisation levels.

3.5 | Problem Definition:

This paper states that, Problem is given as n numbers of workflow tasks $T = \{T_1, T_2, T_3, \ldots, T_n\}$ and a group of m virtual machines $V = \{VM_1, VM_2, VM_3, \ldots, VM_m\}$ the challenge is to identify the task-VM mapping with the following objectives:

- Objective 1: The aggregate makespan (M) is kept to a minimum.
- Objective 2: The aggregate Energy consumption (E) is kept to a minimum.

As considering above objective proposed work is to reduce the amount of time and energy used, as indicated by Eqs. 10, 11, and 12 respectively. So, it makes sense to reduce their linear combination and formulation of the fitness function as follow:

$$Minimize\ z = \alpha \times Makespan + (1-\alpha) \times Energy \tag{8}$$

$$\begin{aligned} Subject\ to\ (i)\ & \sum_{\mathfrak{j}=1}^{\mathfrak{m}} \mathbb{B}_{\mathfrak{i},\mathfrak{j}} = 1, \mathfrak{i} = 1, 2, 3, ..., \mathfrak{n} \\ (ii)\ & 0 \leq \alpha \leq 1 \\ (iii)\ & \alpha = \begin{cases} 0.3 & if\ \ 0.10 \leq \text{CCR} \leq 0.30 \\ 0.5 & if\ \ 0.31 \leq \text{CCR} \leq 0.55 \\ 0.7 & if\ \ 0.56 \leq \text{CCR} \leq 1.00 \end{cases} \end{aligned} \tag{9}$$

The constraint (i) states that each and every workflow task can only be assigned to a single VM, while constraint (ii) limits the alpha (α) range that strikes a balance between makespan and energy and constraint (iii) computed α value that will dependent on Computation to Communication Ratio (CCR) ranges.

3.6 | Problem Formulation:

In order to formulate a bi-objective problem, we must first provide descriptions of the two important parameters, makespan and energy.

Makespan Calculation: Makespan is the actual amount of time required to complete a process. It incorporates the leased $VM's$ *boot time*, data transfer time between two VMs, execution time, and VM *shutdown time*. In order to calculate makespan, it is believed that a virtual machine (VM) cannot run code while information is being transmitted to or from it. Makespan is the

sum of VM *boot time*, the maximum VM*-time* for all VMs, and the VM *shutdown time*. VM*-time*$[i]$ represents a timeframe for the most recent event (beginning from scratch with each and every workflow) at which VM v_i completes the task assigned to it. It's a one-time deal to include the VM *boot time* and the VM *shutdown time* because for the first $VM's$ *boot time* and the last $VM's$ *shutdown time* will make a contribution to makespan, while remaining events will overlap with others. Therefore, makespan can be represented mathematically as:

$$Makespan = VM\text{-}boot\text{-}time + \max_{i=1}^{m} (VM\text{-}time[i]) + VM\text{-}shutdown\text{-}time \tag{10}$$

Energy Calculation: In the Eq. 7, energy model is given where an VM is idle and expected to consume γ W/s. When CPU usage is between 0% to 20%, an extra δ W/s is required for task execution. Similarly, when CPU usage range in between 20% to 50%, the additional energy consumption rises to 3δ W/s. Energy use increases along with the use of the CPU. For example, when a virtual machine's CPU usage is 50%, it uses $\gamma + 5\delta$ W/s. If $gamma = \delta$, then each $delta$ of energy used increases CPU utilisation by 25%. This occurs when the utilisation of the CPU falls below 50%. At 70% CPU utilisation, a virtual machine consumes 3δ W/s, which indicates that each $delta$ of energy expended adds 23.3% CPU usage. Energy efficiency is 20% per δ W/s, 18% per δ W/s, and 16.6% per δ W/s for 80%, 90%, and 100% usage, respectively. Based on the description above, when VM_i is idle, the total energy consumption for task tk_i is determined using the following formula:

$$Total\_Energy = Total\_Energy + \gamma \times (IO\_Time(tk_i) + CPU\_Time(tk_i)) \tag{11}$$

Similarly, when CPU utilization increases, energy consumption will change according to its CPU utilization level. In this case, the total energy consumption for task tk_i is determined using the following formula:

$$Total\_Energy = Total\_Energy + (\gamma + n\delta) \times (IO\_Time(tk_i) + CPU\_Time(tk_i)) \tag{12}$$

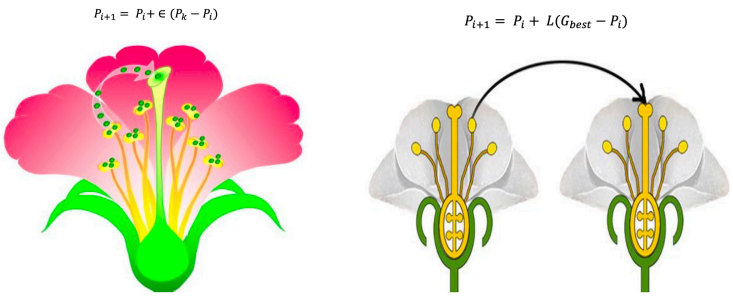
where, $n = (1, 3, 5, 8, 11, 12)$.

4 | PROPOSED APPROACH

In this part, we will first look at a brief overview of flower pollination algorithms and then, demonstrate the pollen representation technique employed in the proposed approach. Following that, we develop a useful multi-objective dynamic fitness function to supplement the suggested approach.

4.1 | Overview of flower pollination algorithm (FPA)

Flower Pollination Algorithm (FPA) is nature-inspired algorithm based on population and proposed by Xin-She Yang[41]. The primary goal of pollinating flowers is to optimise plant reproduction by ensuring the survival of the most robust flowers in flowering plants. The general pollination process is depicted in Figure 3. In this algorithm, self-pollination and cross-pollination



(a) Self-Pollination (b) Cross-Pollination

Figure 3 Pictorial representation of pollination of flowers

are the two important processes. Self-pollination, which frequently occurs when there isn't a reliable pollinator available, occurs

when pollen from the similar flower or different flowers of the same plant is used to fertilize a single flower, such as peach flowers. Cross-pollination, also called allogamy is happens when pollen from a separate plant's flower is used to fertilise another plant's flower. Cross-pollination can occur across great distances. Since, pollinators like bees, bats, birds, and flies are able to fly great distances, they might be regarded as the source of worldwide pollination. The working of FPA is depicted in Figure 4.

4.2 | Fitness Function Evaluation

In order to build an efficient multi-objective dynamic fitness function for the task scheduling problem in cloud computing, we consider two incompatible purposes which are makespan and energy. Using Eqs. 10, 11, and 12, we can always compute the makespan and energy of a given pollen. Let's consider, $Makespan_i$ and $Energy_i$ are the makespan and energy for the i^{th} pollen in population, respectively. The i^{th} pollen's fitness value can now be calculated using Eq. 8.

4.3 | Workflow Scheduling Algorithm Based on FPA

Algorithm 1 provides the pseudo-code for the proposed approach. Step 1 initializes the population of size $pollen\_size$. For each pollen in the present population, the dynamic fitness function value is calculated in steps 2 through 6 using Eq. 8. Note that the values for both energy and makespan are needed for Eq. 8. These values can be computed using Eq. 10, Eq. 11, and Eq. 12. In step 8, the optimal pollen is set up as just an empty set and its fitness value is initialised to the utmost value of the integer in step 5. In steps 9 through 13, the best of the original population's pollen is selected and labelled "Best pollen". In steps 14 through 32, a new random number ranging 0 and 1 is created for each pollen at each iteration. If the derived value exceeds the Switch probability, the pollen is subjected to global pollination using a Levy distribution; otherwise, it is subjected to local pollination using a uniform distribution[41].

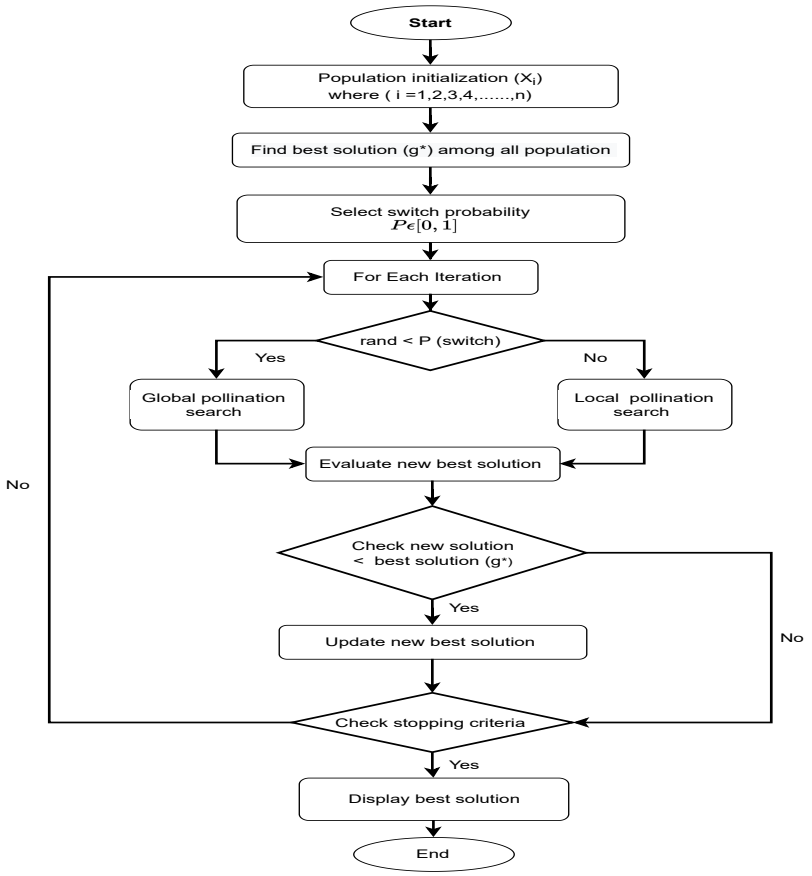


Figure 4 Workflow of Flower Pollination Algorithm

4.4 | Time Complexity Analysis

The Algorithm 1 time complexity is determined by the population size of pollen (n) and the maximum number of iterations (max_iter). The outside loop (lines 14-32) iterates max_iter times, whereas the inner loop (lines 15-31) iterates n times, resulting in an overall time complexity of O(n*max _ter). The Algorithm 2 is a scheduling algorithm that schedules tasks in a topological order and computes the schedule's makespan. The time complexity of the Algorithm 2 is $O(n \times m) + (m \times edge))$ because the outer loop is executed n times, in which n is the overall amount of tasks to be performed, and the inner loop is executed $(m \times edge)$ times where m is workflow and $edge$ is incoming edges in the workflow. Algorithm 3 determines how much energy a group of tasks in a cloud computing environment use collectively. Based on the quantity of actions carried out inside the loop, the time complexity of this algorithm can be examined. The number of tasks is repeated n times in the outer loop. Calculating CPU time, IO time, CPU usage, and updating the total energy are only a few of the tasks performed inside the loop. These operations all have a constant time complexity of $O(1)$. The inner if-else statement runs a constant time of $O(1)$ as well. The time complexity of the Algorithm 3 can be calculated as $O(1)$. Therefore, the total time complexity of the proposed technique is O((max_iter × n)+ (n × m) + (n × edge)), which is likewise the total time complexity of EEWS.

Algorithm 1 Proposed Energy Efficient Workflow Scheduling Algorithm

Require: *No. of* VM_s *(m)*, *pollen_size (n)*, *max_iter*, *P_switch*
Ensure: VM-Task Mapping
1: Initialize pollen $\mathbb{P}_i, 1 \leq i \leq pollen\_size$
2: **for** $i = 1$ to n **do**
3: $Makespan =$ Determine the Makespan's value using Eq. 10
4: $Total_Energy =$ Determine the Energy's value using Eq. 11 and Eq. 12
5: $fit\_i =$ Determine the fitness function's value by j^{th} pollen using Eq. 8
6: **end for**
7: Initialize $\mathbb{G}_{best} = max\_iter$
8: Initialize $\mathbb{B}_{pollen} = \phi$
9: **for** $i = 1$ to n **do**
10: **if** $\mathbb{G}_{best} < fit\_i$ **then**
11: $\mathbb{G}_{best} = \mathbb{P}_i$
12: **end if**
13: **end for**
14: **for** $i = 1$ to max_iter **do**
15: **for** $i = 1$ to n **do**
16: $Temp = Rand[0, 1]$
17: **if** $Temp < P\_switch$ **then**
18: Draw an n-dimensional $\mathbb{L}$ step following a Levy distribution
19: Apply global pollination search using $P_{i+1} = P_i + \mathbb{L}(\mathbb{G}_{best} - P_i)$
20: **else**
21: Draw $\in$ following a Uniform distribution
22: k and j should be chosen randomly from the available solutions
23: Apply local pollination search using $P_{i+1} = P_i + \in (P_k - P_i)$
24: **end if**
25: Calculate new pollen
26: **if** new_pollen $< \mathbb{G}_{best}$ **then**
27: **Update** new_pollen as $\mathbb{G}_{best}$
28: **else**
29: $\mathbb{G}_{best} = \mathbb{B}_{pollen}$
30: **end if**
31: **end for**
32: **end for**

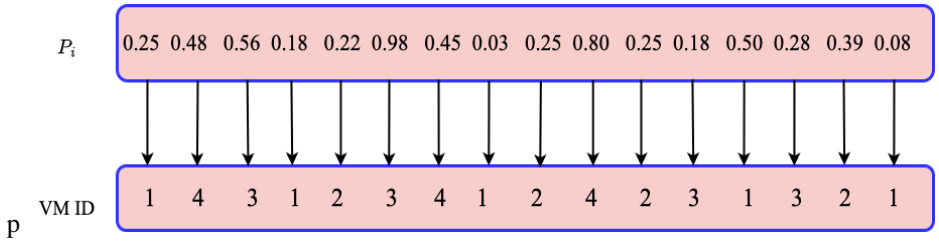


Figure 5 Task-VM mapping extraction from pollen

Algorithm 2 Makespan Calculation

Require: Workflow Application $\mathbb{W}$, Cloud Server Specification $\mathbb{CSS}$, Mapping $\mathbb{M}$
Ensure: Makespan Value (Makespan)

for each $v_i \in \mathbb{V}$ **do**
  $\mathbb{VM}\_time[i] = 0$
end for
for each task $tk_i \in \mathbb{W}$ in topological order **do**
  if $tk_i.Parent\_Count! = 0$ **then**
    $Parent\_Finish\_Time = \max_{tk_k \in pred(tk_i)}(Task\_actual\_finish\_Time[f])$
  end if
  if tk_i.Child_Count $! = 0$ **then**
    Transfer_Time $= 0$
    for each task tk_j where $tk_j \in succ(tk_i)$ and $\mathbb{M}[i] \neq \mathbb{M}[j]$ **do**
      if output data of tk_i task is not transferred to $v_{\mathbb{M}[j]}$ **then**
        $Transfer\_Time = Transfer\_Time + \frac{tk_i.Output\_Data\_Size}{Bandwidth}$
      end if
    end for
  end if
  $Execution\_Time\ ET_{tk_i}^{v_{\mathbb{M}[i]}} = \frac{Load(tk_i)}{Capacity(v_{\mathbb{M}[i]}) \times (1 - deg_{v_{\mathbb{M}[i]}})}$
  $Actual\_Start\_Time = max(Parent\_Finish_Time, \mathbb{VM}\_time[i])$
  $Task\_Actual\_Finish\_Time[i] = Actual\_Start\_Time + Execution\_Time + Transfer\_Time$
  $\mathbb{VM}\_time[\mathbb{M}[i]] = Task\_Actual\_Finish\_Time[i]$
end for
$Makespan = \mathbb{VM}\_Boot\_Time + \max_{v_i \in Cloud}(\mathbb{VM}\_time[i]) + \mathbb{VM}\_Shutdown\_Time$
return Makespan

4.5 | An illustration

Figure 6 represents an illustration of DAG workflow where nodes stand for actual tasks, while the edges show the priority relationship between them. In node T_4, the number 20 represents the computational load of task T_4, and the edge value 35 represents the amount of data transferred from T_4 to T_{10}. $E_k = (T_i, T_j)$ is an edge which represents the precedence relationship among task T_i and T_j. In other words, if an edge $E_k = (T_i, T_j)$ exists in a given set E, then task T_i is predecessor to task T_j and task execution T_j can start after task T_i is completed. As a result, the earliest start time ($\mathbb{EST}$) of a task T_j is calculated by multiplying the actual finish time ($\mathbb{AFT}$) of all its ancestors with the formula 13.

$$\mathbb{EST}(T_j) = \max_{T_i\ \epsilon\ pred\ (T_j)} \left\{ \mathbb{AFT}(T_i) \right\} \tag{13}$$

Algorithm 3 Energy Calculation

Require: Workflow Application $\mathbb{W}$, Cloud Server Specification $\mathbb{CSS}$, Mapping $\mathbb{M}$
Ensure: Energy Value (Total_Energy)

Set Total_Energy = 0
for each task $tk_i \in \mathbb{W}$ **do**
  $CPU\_Time(tk_i) = \frac{Computation\_Load(tk_i)}{Speed\_of\_\mathbb{VM}_j}$
  $IO\_Time(tk_i) = \frac{F\_Total}{IO\_Speed}$
  $CPU\_Util(\mathbb{VM}_j) = \frac{CPU\_Time(tk_i)}{(IO\_Time(tk_i)+CPU\_Time(tk_i))}$
  if $0 < CPU\_Util(\mathbb{VM}_j) \leq 0.2$ **then**
    $Total\_Energy = Total\_Energy + (\gamma + \delta) \times (IO\_Time(tk_i) + CPU\_Time(tk_i))$
  else if $0.2 < CPU\_Util(\mathbb{VM}_j) \leq 0.5$ **then**
    $Total\_Energy = Total\_Energy + (\gamma + 3\delta) \times (IO\_Time(tk_i) + CPU\_Time(tk_i))$
  else if $0.5 < CPU\_Util(\mathbb{VM}_j) \leq 0.7$ **then**
    $Total\_Energy = Total\_Energy + (\gamma + 5\delta) \times (IO\_Time(tk_i) + CPU\_Time(tk_i))$
  else if $0.7 < CPU\_Util(\mathbb{VM}_j) \leq 0.8$ **then**
    $Total\_Energy = Total\_Energy + (\gamma + 8\delta) \times (IO\_Time(tk_i) + CPU\_Time(tk_i))$
  else if $0.8 < CPU\_Util(\mathbb{VM}_j) \leq 0.9$ **then**
    $Total\_Energy = Total\_Energy + (\gamma + 11\delta) \times (IO\_Time(tk_i) + CPU\_Time(tk_i))$
  else if $0.9 < CPU\_Util(\mathbb{VM}_j) \leq 1$ **then**
    $Total\_Energy = Total\_Energy + (\gamma + 12\delta) \times (IO\_Time(tk_i) + CPU\_Time(tk_i))$
  else
    $Total\_Energy = Total\_Energy + \gamma \times (IO\_Time(tk_i) + CPU\_Time(tk_i))$
  end if
end for
return $Total\_Energy$

Similarly, to compute the makespan of a given DAG, determine the amount of time required to complete the execution of last task T_{exit} as shown in the Eq. 14.

$$M = \mathbb{AFT}(T_{exit}) \tag{14}$$

To calculate the execution time of specified tasks, performance variability of the virtual machine VM_K and computational speed are computed and represented by D and S_{VM_K} respectively. The total number of instructions in MI necessary to execute a certain task T_i is represented by each node's weight, WT_i. Thus, the time required to execute task T_i on VM_k is shown below in Eq. 15, taking into account the decrease in voltage supply by dVi and the relative decrement in VM speed as $spdV_i$.

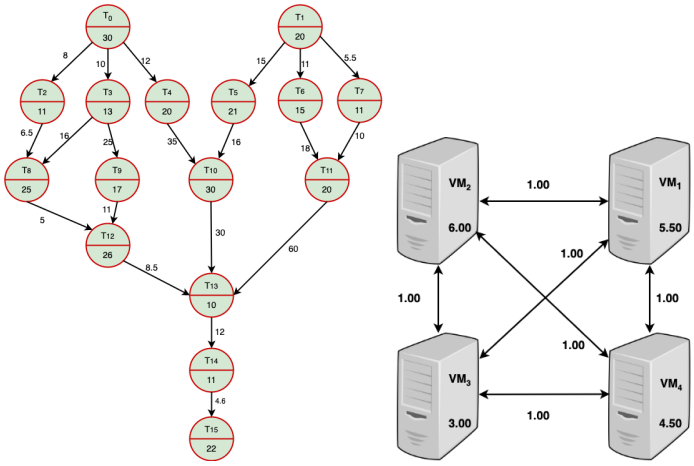
$$ET_{T_i}^{VM_k} = \frac{W_{T_i}}{(S_{VM_k} \times (1 - D - spdV_i))} \tag{15}$$

Consider a Montage workflow with 16 tasks in it, where $T = \{T_1, T_2, T_3, ..., T_{16}\}$ and a group of virtual machines, $V = \{VM_1, VM_2, VM_3, VM_4\}$ as described in Figure 6. On the specified VMs, which are fully connected to one another, we must arrange the workflow. The result of this example is a mapping of specific tasks to virtual machines that saves both time and energy. The parameters used in this demonstration are shown in Table 3, and the working of the FPA is depicted in Figure 4.

An n-dimensional vector (p) is used to represent pollen, where each pollen element ($p_{ij} \mid 1 \leq i, j \leq n$) is having a random number between (0, 1), i.e., $0 < p_{ij} < 1$. Eq. 16 provides the mapping function that is being used.

$$Task(VM_{ij}) = floor(p_{ij} \times m) + 1 \mid 1 \leq i \leq p_{size} \text{ and } i \leq j \leq n \tag{16}$$

where, p_{size} for population size, m stands for the overall number of virtual machines, and n for the overall number of tasks. It is noteworthy that each pollen can produce an entire answer to the task scheduling issue. We use an imaginary could-resource



(a) Workflow montage containing 16 tasks. (b) The cloud environment

Figure 6 The workflow and cloud environment

model with n = 16 and m = 4 to illustrate this. By taking a look at the sampled pollen in Figure 5, the task-VM mapping procedure is demonstrated.

Table 3 The parameters utilized in illustration.

Parameters	**Value**
Computing power of all VMs	5.5, 6.0, 3.0 and 4.5 $MIPS$
Number of VMs	4
Network bandwidth	1 $MBps$
Population size (N)	200
Max iterations	10
Boot time and shutdown time of VM	0.5 sec
Mass threshold for inferior pollens (δ)	0.05
Weight of makespan and energy (α)	0.5
Small constant used in gravity (γ)	0.1
Energy time equivalence (β)	1

In Figure 5, it can be seen that the initial task P_1 is given to the 1's virtual machine with an assign value of 0.25 which results in floor $(0.25 \times 4) + 1 = 1$. Similar steps are taken to map the remaining tasks to VMs. For example, using the same approach from Eq. 16, the tasks T_2 and T_3 are mapped into VM_4 and VM_3. The task and virtual machine mapping are shown in Figure 5. We compute the fitness function after mapping. To calculate the fitness function consider FPA's operation, as illustrated in Figure 4. Initially, evaluate the values of each pollen and selected $n = 9$ for population size. Each pollen is assigned a value based on a uniform random distribution with a range of [0, 1]. In Figure 7, demonstrates how the pollen values are initialised using a uniform distribution. After the initialization of pollen values, the g_best values were selected from the randomly generated values. Using the same distribution, a second random value will be generated. This random value is compared to the value of the static p_switch. If the random value is smaller than the p_switch value, a global pollination search is performed; otherwise, a local pollination search is conducted. This will result in the acquisition of a set of values. The newly discovered value will now be compared to the g_best value that was chosen previously. The new search value will be changed to g_best if the selected number is less than g_best; otherwise, nothing will change. Using Eq. 8, determine the fitness function. In order to calculate the fitness function, first determine the makespan and energy. Makespan and energy can be determined using Algorithms 2 and 3.

For the particular workflow depicted in Figure 6, we have determined the makespan, energy, and fitness value. The population size in this case is 200. With a threshold value of 0.05, we performed the experiment for 10 iterations. Table 4 displays the experimental iteration-wise computation of makespan for four different algorithms.

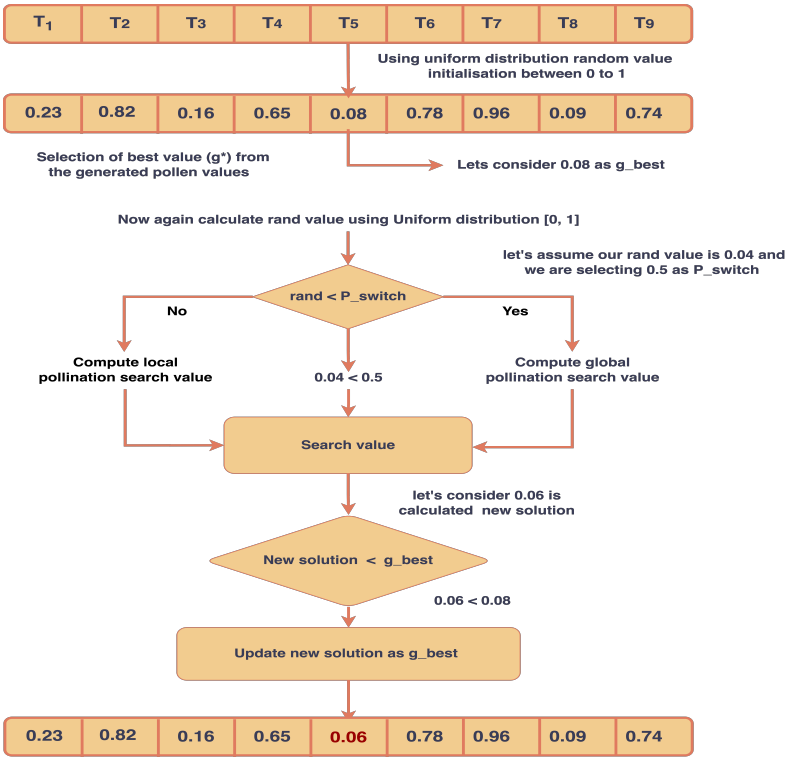


Figure 7 Simple calculations on FPA

Table 4 Iteration-wise Specification of the Makespan.

Iteration	Makespan of FPA	Makespan of GA	Makespan of GSA	Makespan of PSO
1	3.656E+03	7.678E+03	8.885E+03	7.299E+03
2	3.446E+03	6.283E+03	3.743E+03	4.519E+03
3	4.512E+03	3.841E+03	2.915E+03	8.806E+03
4	3.588E+03	7.131E+03	5.398E+03	2.688E+03
5	3.841E+03	5.168E+03	4.131E+03	5.578E+03
6	3.533E+03	3.737E+03	5.175E+03	3.420E+03
7	2.917E+03	5.804E+03	3.324E+03	6.720E+03
8	3.450E+03	3.522E+03	4.416E+03	5.663E+03
9	4.073E+03	3.379E+03	4.564E+03	6.444E+03
10	3.520E+03	3.477E+03	3.578E+03	3.529E+03

At iteration 1, the makespan values for the FPA, GA, GSA, and PSO are $3.656E+03$, $7.678E+03$, $8.885E+03$, and $7.299E+03$ respectively. We can observe that FPA has the lowest makespan value among all algorithms. Similarly, when we repeat the iteration, we get the same result as a low FPA makespan value. We calculated the values for energy and fitness in the same way as Makepan and received the same results. The iteration-by-iteration calculation for energy and fitness values is shown in Tables 5 and 6. We compared the FPA to other algorithms such as GA, PSO, and GSA and discovered that the FPA produces good results when compared to other algorithms.

This is because FPA has extremely few parameters and has demonstrated robust performance in a variety of optimization tasks. Furthermore, FPA is a robust, flexible, scalable, and straightforward optimization method. As a result, when compared to other metaheuristic algorithms, FPA produces good results for addressing a variety of real-world optimization problems.

Table 5 Iteration-wise Specification of the Energy.

Iteration	Energy of FPA	Energy of GA	Energy of GSA	Energy of PSO
1	10.360E+03	13.790E+03	13.118E+03	10.356E+03
2	3.4190E+03	10.522E+03	3.8441E+03	10.423E+03
3	10.450E+03	10.528E+03	13.620E+03	10.528E+03
4	6.5270E+03	7.1631E+03	10.521E+03	10.531E+03
5	7.2661E+03	13.705E+03	9.8321E+03	9.9490E+03
6	9.7862E+03	10.364E+03	13.710E+03	13.002E+03
7	10.425E+03	10.422E+03	10.342E+03	9.8443E+03
8	3.4191E+03	6.5261E+03	10.350E+03	9.8714E+03
9	3.2682E+03	10.354E+03	16.957E+03	7.1766E+03
10	3.2573E+03	10.422E+03	16.395E+03	9.7723E+03

Table 6 Iteration-wise Specification of the Fitness.

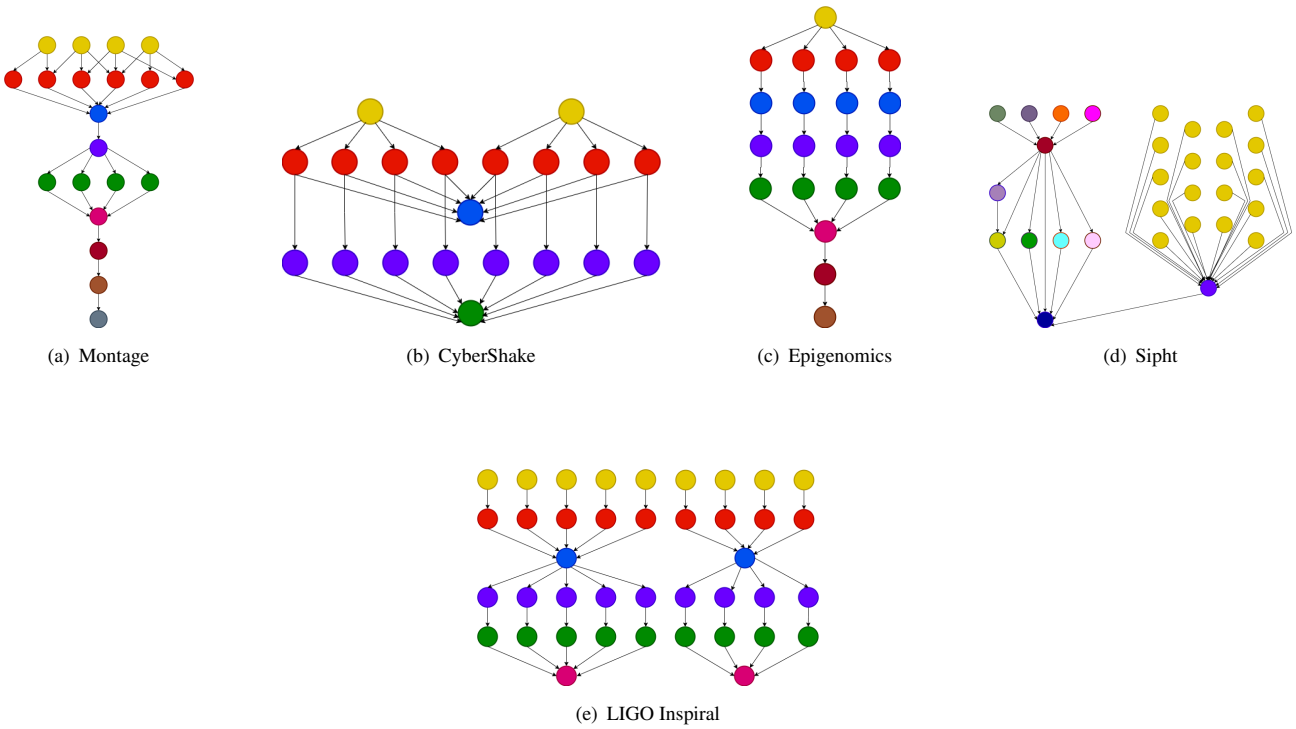
Iteration	Fitness of FPA	Fitness of GA	Fitness of GSA	Fitness of PSO
1	7.008E+03	1.073E+04	1.100E+04	8.828E+03
2	3.432E+03	8.402E+03	3.793E+03	7.471E+03
3	7.481E+03	7.184E+03	8.268E+03	9.667E+03
4	5.058E+03	7.147E+03	7.959E+03	6.610E+03
5	5.553E+03	9.437E+03	6.982E+03	7.763E+03
6	6.660E+03	7.051E+03	7.143E+03	8.211E+03
7	6.671E+03	8.113E+03	6.833E+03	8.282E+03
8	3.435E+03	5.024E+03	7.383E+03	7.767E+03
9	3.671E+03	6.866E+03	1.076E+04	6.810E+03
10	3.391E+03	6.950E+03	9.987E+03	6.651E+03

5 | EXPERIMENTAL RESULTS

In this section, simulation results are shown, as well as comparisons between the proposed method and three other state-of-the-art algorithms that already exist, namely GA, PSO, and GSA. The proposed FPA algorithm was evaluated using simulations executed on a MacBook Pro with 512 GB of SSD storage and 16 GB of RAM, running macOS Ventura 13.0.1 and spyder 5.3.3. Various workflows combinations was selected based on CCR values. CCR is the ratio of average communication expense to the average computation expense. Higher CCR values are associated with workflows that are data-intensive, while lower CCR values are associated with workflows that are compute-intensive. So, if a process is compute intensive, it requires a great deal of energy to function; similarly, if a workflow is data intensive, it requires a significant amount of time to complete. Based on this, we calculate alpha value using the CCR value indicated in Eq. 9.

5.1 | Datasets Used

For the simulations, we took into account five different types of scientific workflows[42], namely Montage (network-intensive), LIGO Inspiral (compute-intensive), Cybershake (IO and network-intensive), Epigenomic (both compute-intensive and network-intensive), and Sipht (IO intensive), each of which has 24-2000 nodes of task. The fundamental workflow structure of all datasets is depicted in Figure 8. Epigenomic and SIPHT datasets are used in biology, whereas the LIGO Inspiral dataset is used in gravitational physics and the Montage dataset is used in astronomy.



(a) Montage (b) CyberShake (c) Epigenomics (d) Sipht

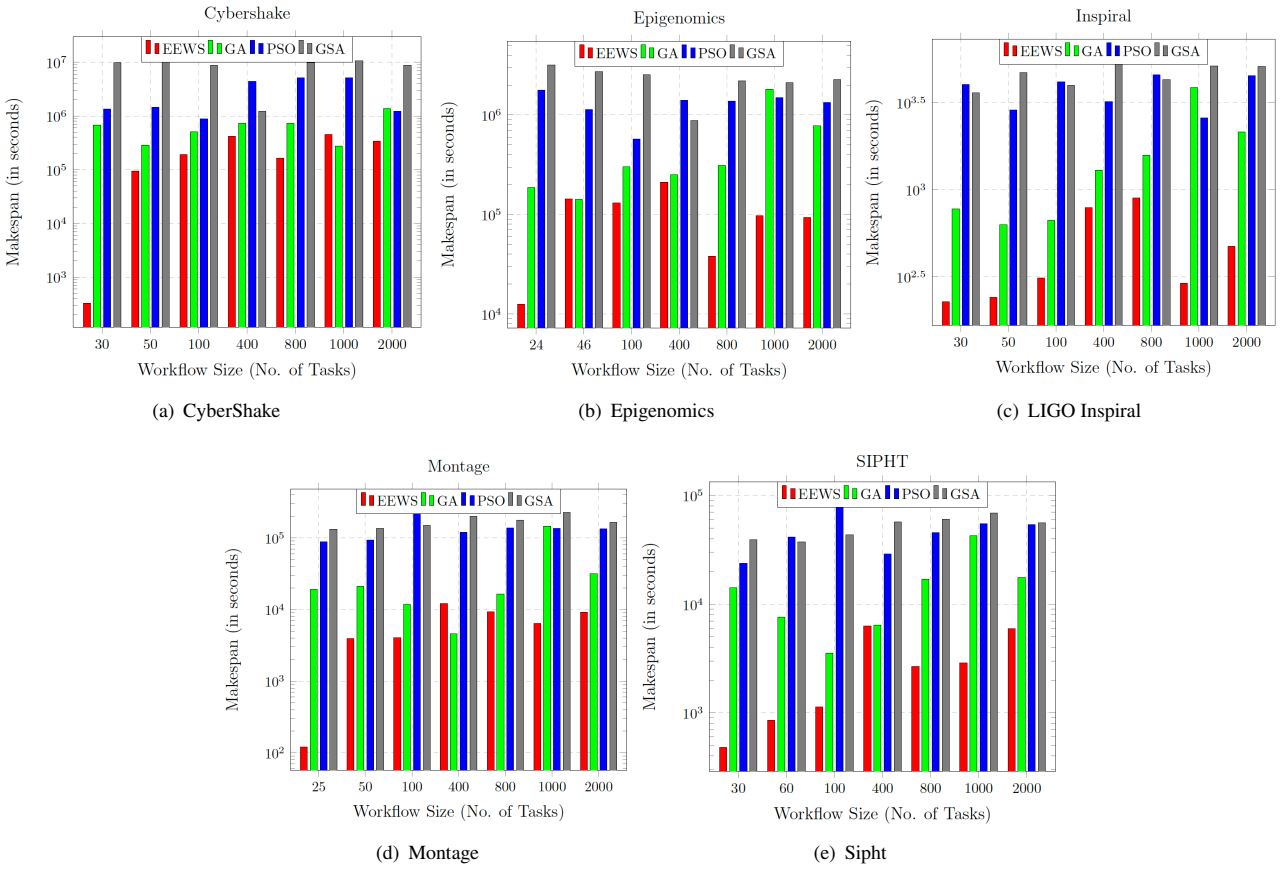
(e) LIGO Inspiral

Figure 8 Datasets

5.2 | Analysis of results and performance evaluation

In this part of the article, we will compare the efficiency of our proposed algorithm with that of GA, GSA, and PSO in terms of both makespan and energy. Figure 9 depicts the compression of Makesapn using various alorithms. In Figure 9(a), we compare the Cybershake dataset to various algorithms, including GA, PSO, GSA, and the proposed methodology. The task count is shown on the x-axis, and the task execution time is shown in seconds on the y-axis. We considered a minimum of 30 and a maximum of 2000 tasks for this dataset. We used specific colours to represent various algorithms, such as red for the proposed methodology (EEWS).

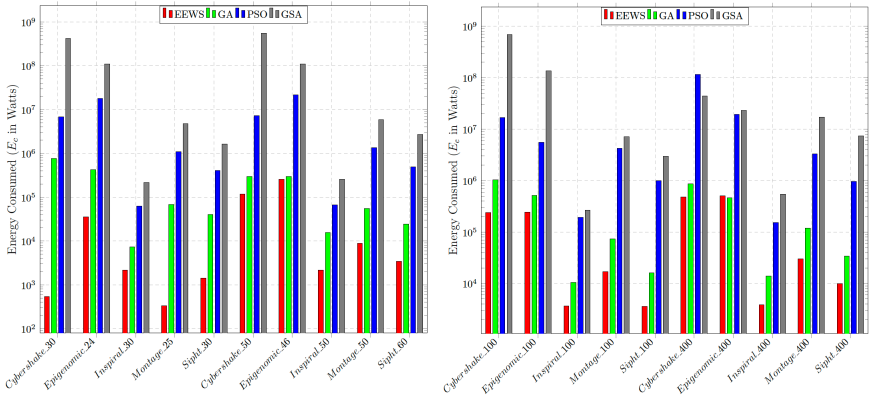
The GA, PSO, and GSA algorithms are represented by the colours green, blue, and grey respectivily. The primary goal is to narrow the makespan. We can see that the proposed algorithms take less time to execute tasks when compared to other algorithms. Consider Figure 9(c), where we used the Inspiral dataset and compared it to various algorithms. If we look at Figure 9(c), we can see that different tasks are defined. Let us consider task 400. If we look at the time it takes to complete the task execution for GA, it is approximately $10^{3.1}$ seconds. Similarly, PSO and GSA will take $10^{3.5}$ sec and $10^{3.4}$ sec respectively. If we consider our proposed approach, it takes less than $10^{3.0}$ sec. The result, which is clearly shown in Figure 9(c), is the same no matter how many tasks we check for. Similarly, if we consider the Epigenomics dataset depicted in Figure 9(b). The makespan for 46 tasks is the same for the proposed method and GA algorithms, which is $10^{5.2}$, but PSO and GSA have taken longer makespans, $10^{6.0}$ and $10^{6.5}$, respectively. If we consider the Montage dataset containing 400 tasks, as depicted in Figure 9(d), the makespan required by the proposed algorithm is longer than that of other algorithms. Similarly, if we examine the SIPHT dataset containing 400 tasks, the makespan difference between the proposed algorithm and the GA presented in Figure 9(e) is minimal. The same experiment we ran for other datasets is shown in Figure 9. For each dataset, we obtained the mostly same conclusion. Minimizing energy usage is the second goal of the proposed approach. To do this, we conducted a number of related experiments using the various datasets depicted in Figure 10. The proposed approach performed well when we compared the results to those of other algorithms like PSA, GA, and GSA. Here, we have utilised the same colour representation for the method that was explained earlier. There are two distinct configurations is use in Figure 10 where Set 1 contains a small number of tasks (24 to 60) and Set 2 contains a medium number of tasks (100 to 400). Set 3 contains a large number of tasks (800 to 2,000) which is shown in Figure 11. All sets are used for comperison. With several datasets sets plotted on the x-axis and energy consumption in watts on the y-axis, entire set represents the energy consumption graph. Let's take *set*_2 into consideration, which represents



(a) CyberShake (b) Epigenomics (c) LIGO Inspiral

(d) Montage (e) Sipht

Figure 9 Makespan comparison for (a) Cybershake, (b) Epigenomic, (c) Inspiral, (d) Montage and (e) SIPHT



(a) Set_1 (b) Set_2

Figure 10 Comparison of Energy Consumption for Set_1, Set_2

the dataset name as $Sipht\_100$. This indicates that the experiment was conducted on the Sipht dataset, and 100 tasks were used to compute the energy usage. We followed the same naming convention for each dataset, which is dataset name followed by the number of tasks. Again, we have chosen minimum 24 and maximum 2000 tasks for the experiments, and in each instance, the proposed approach is outperforming other algorithms.

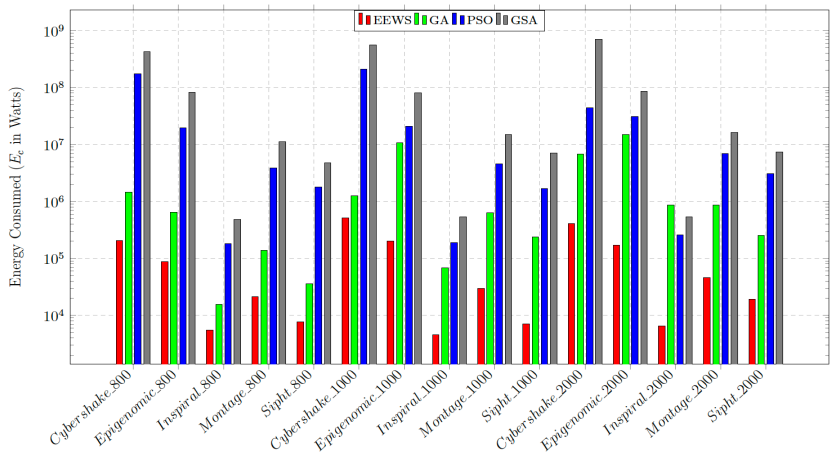


Figure 11 Comparison of Energy Consumption for *Set*_3

Let's look at the dataset *montage*_400. We can observe that the proposed approach consumes about $10^{4.5}$ *watts* energy. Similar results are obtained when the same problem is computed using various algorithms: $10^{5.1}$ *watts* for GA, $10^{7.1}$ *watts* for GSA, and $10^{6.8}$ *watts* for PSO. Similarly, if we examine *Epigenomic*_400, we can observe that the proposed method uses slightly more energy than GA. If we consider *set*_1 and compare the energy utilities by *Epigenomic*_46 to GA and the proposed method, the results are nearly identical. In *set*_3 as demonstrated by comparisons between proposed algorithms and other algorithms, the proposed approach performs well. From the comparison of makespan and energy utilities, we can see that when the number of tasks is high, the proposed algorithm performs well. For *Epigenomic* dataset with a limited number of tasks, the proposed algorithm does not produce satisfactory outcomes.

5.3 | Analysis of variance (ANOVA)

For computation, we employed five datasets, and one-way ANOVA was used for statistical analysis. Here, the rationale for employing One-way ANOVA is covered. The Montage dataset includes images of various astronomical objects, and ANOVA can be used to compare the mean pixel intensities of different objects to see if there are any significant differences in brightness or contrast. The CyberShake dataset contains seismic hazard information for the Los Angeles area, and ANOVA can be used

Table 7 ANOVA analysis for CyberShake workflow.

Makespan						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	2.03E+14	3	6.78E+13	10.0817	4.46E-05	2.838
Within Groups	2.69E+14	40	6.73E+12			
Total	4.72E+14	43				
Energy						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	7.21E+17	3	2.40E+17	9.99175	4.81E-05	2.838
Within Groups	9.62E+17	40	2.05E+16			
Total	1.68E+18	43				
Fitness Fuction						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	1.86E+17	3	6.20E+16	10.0155	4.71E-05	2.838
Within Groups	2.480E+17	40	6.20E+15			
Total	4.34E+17	43				

to compare the mean seismic hazard values in different regions to see if there are any significant differences in hazard levels. The Epigenomics dataset contains information about DNA epigenetic modifications in various tissues, and ANOVA can be used to compare the mean levels of epigenetic modifications in various tissues to see if there are any significant differences in the epigenetic landscape.

Table 8 ANOVA analysis for Epigenomic workflow.

Makespan						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	1.15E+13	3	3.86E+12	7.05885	0.64E-03	2.838
Within Groups	2.18E+13	40	5.46E+11			
Total	3.34E+13	43				
Energy						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	2.37E+16	3	7.92E+15	11.0354	2.05E-05	2.838
Within Groups	2.87E+16	40	7.18E+14			
Total	5.24E+16	43				
Fitness Fuction						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	6.19E+15	3	2.06E+15	10.9840	2.14E-05	2.838
Within Groups	7.52E+15	40	1.88E+14			
Total	1.37E+16	43				

The Sipht dataset contains information about the protein folding process, and ANOVA can be used to compare the mean folding rates of different proteins to see if there are any significant differences. The Inspiral dataset contains information about binary black holes' inspiral phases, and ANOVA can be used to compare the mean inspiral waveforms of different binary systems to see if there are any significant differences in inspiral behaviour. In all of these cases, ANOVA allows you to compare the means of different groups in a dataset and determine whether the differences are significant or if they could have happened by chance.This test consists of a null hypothesis (H_0) and an alternative hypothesis (H_1), which are defined as follows:

$$H_0 : \mu_1 = \mu_2 = \mu_3 = ... = \mu_n \tag{17}$$

$$H_1 : Means\ are\ not\ equal \tag{18}$$

Table 9 ANOVA analysis for Inspiral workflow.

Makespan						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	4.74E+7	3	1.58E+7	6.9517	7.11E-04	2.838
Within Groups	9.10E+7	40	2.27E+6			
Total	1.38E+8	43				
Energy						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	4.51E+11	3	1.50E+11	9.3390	8.34E-05	2.838
Within Groups	6.43E+11	40	1.61E+10			
Total	1.09E+12	43				
Fitness Fuction						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	1.16E+11	3	3.8E+10	9.6341	6.49E-05	2.838
Within Groups	1.61E+11	40	4.63E+9			
Total	2.78E+11	43				

If $F\_statistical$ is greater than $F\_critical$ during the test, we cannot reject the null hypothesis and all groups have the same mean. In contrast, if $F\_statistical$ exceeds $F\_critical$, the null hypothesis is refuted and the alternative hypothesis is accepted. If the alternative hypothesis is accepted, it is straightforward to conclude that one of the categories has significant differences from the others.

Table 10 ANOVA analysis for Montage workflow.

Makespan						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	7.26E+10	3	2.42E+10	6.4542	1.14E-03	2.838
Within Groups	1.50E+11	40	3.75E+09			
Total	2.22E+11	43				
Energy						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	3.55E+14	3	1.18E+14	9.09868	1.02E-04	2.838
Within Groups	5.20E+14	40	1.30E+13			
Total	8.75E+14	43				
Fitness Fuction						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	9.01E+13	3	3.00E+13	8.91487	1.20E-04	2.838
Within Groups	1.34E+14	40	3.37E+12			
Total	2.24E+14	43				

The test was conducted to compare the GSA, GA, PSO, and FPA standards. In this investigation, each of the four algorithms was executed 11 times for each of the five scientific workflows of differing sizes with $\alpha = 0.05$. Tables [7 - 11] displays the outcomes for each workflow.

Table 11 ANOVA analysis for Sipht workflow.

Makespan						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	8.25E+09	3	2.75E+09	7.81331	3.20E-04	2.838
Within Groups	1.40E+10	40	3.52E+08			
Total	2.23E+10	43				
Energy						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	6.92E+13	3	2.30E+13	8.8303	1.29E-04	2.838
Within Groups	1.04E+14	40	2.61E+12			
Total	1.73E+14	43				
Fitness Fuction						
Source of Variation	SS	df	MA	F	p-value	F crit
Between Groups	1.76E+13	3	5.87E+12	8.8436	1.27E-04	2.838
Within Groups	2.65E+13	40	6.64E+11			
Total	4.42E+13	43				

6 | CONCLUSION AND FUTURE WORK

In this paper, we have proposed a task scheduling algorithm that is capable to schedule workflow application in cloud computing infrastructure. The proposed algorithm have been modelled by pollination of flower known as Flower Pollination Algorithm

(FPA). The proposed algorithm has shown that a pollen can be represented effectively to acquire the mapping between tasks and virtual machines as targeting the performance criteria, makespan and energy consumption. The proposed fitness function is dynamic in its nature becaues of computation to communication ratio (CCR). The proposed algorithm has been illustrated with an example and the time complexity have been described precisely. The proposed algorithm have been evaluated using five benchmark datasets and compared it to three well-known algorithms: PSO, GA, and GSA. According to the results of our simulations, the proposed algorithm outperforms the other three algorithms in terms of makespan and energy utilisation. In addition, we have performed a one-way analysis of variance to confirm the findings of our study. Future work will involve the creation of a container-based energy-efficient technology. In addition, a container migration strategy will be devised to reduce energy usage without compromising service quality.

ACKNOWLEDGMENTS

Author contributions

Each author's contribution is equal.

Financial disclosure

None reported.

Conflict of interest

The authors declare no potential conflict of interests.

SUPPORTING INFORMATION

There is no supporting information is available as part of the online article:

References

1. Iranmanesh A, Naji HR. DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing. *Cluster Computing* 2021; 24: 667–681.

2. Kakkottakath Valappil Thekkepuryil J, Suseelan DP, Keerikkattil PM. An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment. *Cluster Computing* 2021; 24: 2367–2384.

3. Belgacem A, Beghdad-Bey K. Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost. *Cluster Computing* 2022; 25(1): 579–595.

4. Garg N, Singh D, Goraya MS. Energy and resource efficient workflow scheduling in a virtualized cloud environment. *Cluster Computing* 2021; 24: 767–797.

5. Katal A, Dahiya S, Choudhury T. Energy efficiency in cloud computing data centers: a survey on software technologies. *Cluster Computing* 2022: 1–31.

6. Rodriguez MA, Buyya R. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurrency and Computation: Practice and Experience* 2017; 29(8): e4041.

7. Chen L, Li X, Guo Y, Ruiz R. Hybrid resource provisioning for cloud workflows with malleable and rigid tasks. *IEEE Transactions on Cloud Computing* 2019; 9(3): 1089–1102.

8. Kamiya G. Data Centres and Data Transmission Networks. September 2022. https://www.iea.org/reports/data-centres-and-data-transmission-networks [Accessed: (Use the date of access)].

9. Bharany S, Badotra S, Sharma S, et al. Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy. *Sustainable Energy Technologies and Assessments* 2022; 53: 102613.

10. Kaur S, Kumar Y, Koul A, Kumar Kamboj S. A Systematic Review on Metaheuristic Optimization Techniques for Feature Selections in Disease Diagnosis: Open Issues and Challenges. *Archives of Computational Methods in Engineering* 2022: 1–33.

11. Xu X, Dou W, Zhang X, Chen J. EnReal: An energy-aware resource allocation method for scientific workflow executions in cloud environment. *IEEE transactions on cloud computing* 2015; 4(2): 166–179.

12. Ahmad SG, Liew CS, Munir EU, Ang TF, Khan SU. A hybrid genetic algorithm for optimization of scheduling workflow applications in heterogeneous computing systems. *Journal of Parallel and Distributed Computing* 2016; 87: 80–90.

13. Hai T, Zhou J, Jawawi D, et al. Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes. *Journal of Cloud Computing* 2023; 12(1): 15.

14. Shishira S, Kandasamy A, Chandrasekaran K. Workload scheduling in cloud: A comprehensive survey and future research directions. In: IEEE. ; 2017: 269–275.

15. Lee YC, Zomaya AY. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing* 2012; 60(2): 268–280.

16. Askarizade Haghighi M, Maeen M, Haghparast M. An energy-efficient dynamic resource management approach based on clustering and meta-heuristic algorithms in cloud computing IaaS platforms: Energy efficient dynamic cloud resource management. *Wireless Personal Communications* 2019; 104: 1367–1391.

17. Srikantaiah S, Kansal A, Zhao F. Energy aware consolidation for cloud computing. In: . 10. San Diego, California. ; 2008: 1–5.

18. Usman MJ, Gabralla LA, Aliyu A, Gabi D, Chiroma H. Multi-Objective Hybrid Flower Pollination Resource Consolidation Scheme for Large Cloud Data Centres. *Applied Sciences* 2022; 12(17): 8516.

19. Kalra M, Singh S. A review of metaheuristic scheduling techniques in cloud computing. *Egyptian informatics journal* 2015; 16(3): 275–295.

20. Rodriguez MA, Buyya R. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing* 2014; 2(2): 222–235.

21. Schad J, Dittrich J, Quiané-Ruiz JA. Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proceedings of the VLDB Endowment* 2010; 3(1-2): 460–471.

22. Panda SK, Jana PK. Efficient task scheduling algorithms for heterogeneous multi-cloud environment. *The Journal of Supercomputing* 2015; 71(4): 1505–1533.

23. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems* 2009; 25(6): 599–616.

24. Singh V, Gupta I, Jana PK. A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources. *Future Generation Computer Systems* 2018; 79(Part 1): 95 - 110. doi: https://doi.org/10.1016/j.future.2017.09.054

25. Lee YC, Han H, Zomaya AY, Yousif M. Resource-efficient workflow scheduling in clouds. *Knowledge-Based Systems* 2015; 80: 153–162.

26. Vasile MA, Pop F, Tutueanu RI, Cristea V, Kołodziej J. Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. *Future Generation Computer Systems* 2015; 51: 61–71.

27. Hsu CH, Slagter KD, Chen SC, Chung YC. Optimizing energy consumption with task consolidation in clouds. *Information Sciences* 2014; 258: 452–462.

28. Kim N, Cho J, Seo E. Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems. *Future Generation Computer Systems* 2014; 32: 128–137.

29. Wen G, Hong J, Xu C, Balaji P, Feng S, Jiang P. Energy-aware hierarchical scheduling of applications in large scale data centers. In: IEEE. ; 2011: 158–165.

30. Usman MJ, Ismail AS, Chizari H, et al. Energy-efficient virtual machine allocation technique using flower pollination algorithm in cloud datacenter: a panacea to green computing. *Journal of Bionic Engineering* 2019; 16(2): 354–366.

31. Tran MN, Kim Y. A Cloud QoS-driven Scheduler based on Deep Reinforcement Learning. In: IEEE. ; 2021: 1823–1825.

32. Bloch TA, Rajagopal S, Ranga PC. IAGA: Interference Aware Genetic Algorithm based VM Allocation Policy for Cloud Systems. *International Journal of Advanced Computer Science and Applications* 2022; 13(4).

33. Abd Elaziz M, Xiong S, Jayasena K, Li L. Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. *Knowledge-Based Systems* 2019; 169: 39–52.

34. Bacanin N, Bezdan T, Tuba E, Strumberger I, Tuba M, Zivkovic M. Task scheduling in cloud computing environment by grey wolf optimizer. In: IEEE. ; 2019: 1–4.

35. Chen X, Cheng L, Liu C, et al. A WOA-based optimization approach for task scheduling in cloud computing systems. *IEEE Systems Journal* 2020; 14(3): 3117–3128.

36. Shukri SE, Al-Sayyed R, Hudaib A, Mirjalili S. Enhanced multi-verse optimizer for task scheduling in cloud computing environments. *Expert Systems with Applications* 2021; 168: 114230.

37. Velliangiri S, Karthikeyan P, Xavier VA, Baswaraj D. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Engineering Journal* 2021; 12(1): 631–639.

38. Hu N, Tian Z, Du X, Guizani N, Zhu Z. Deep-green: a dispersed energy-efficiency computing paradigm for green industrial IoT. *IEEE Transactions on Green Communications and Networking* 2021; 5(2): 750–764.

39. Attiya I, Abd Elaziz M, Xiong S. Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm. *Computational intelligence and neuroscience* 2020; 2020.

40. Hussain M, Wei LF, Lakhan A, Wali S, Ali S, Hussain A. Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustainable Computing: Informatics and Systems* 2021; 30: 100517.

41. Yang XS. Flower pollination algorithm for global optimization. In: Springer. ; 2012: 240–249.

42. Deelman E, Gannon D, Shields M, Taylor I. Workflows and e-Science: An overview of workflow system features and capabilities. *Future generation computer systems* 2009; 25(5): 528–540.

AUTHOR BIOGRAPHY

Sahani Pooja Jaiprakash has completed her M. Tech. in Computer Engineering from Uka Tarsadia University, Bardoli, India in 2019. She has received her BE from SCET, India in 2017. She is currently pursuing her PhD. from Bennett University

and assistant proffesor at Institute of Technical Education Research (ITER), Siksha 'O' Anusandhan (Deemed to be University. Her research interests includes Images forensics, Image processing, Pattern Recognization, Machine Learning, Deep Learning and Cloud Computing.

Harsh Kumar Arya has completed his B.Tech from Bennett University. His research interests is Cloud Computing.

Dr.Indrajeet Gupta received Ph.D. degree in Computer Science & Engineering from the Indian Institute of Technology (ISM) Dhanbad, India, in 2019 and MTech. from NIT Rourkela, India, in 2012. Currently, he is an educator serving as an Associate Professor in the School of CSET at Bennett University, Greater Noida, India. He is Amazon Web Service (AWS) Educate Faculty Brand Ambassador since 2019. He is also AWS accredited cloud practitioner and Microsoft Azure certified Educator of the MS Learn Program powered by Microsoft. His interest areas include workflow scheduling, Cloud Resource Provisioning, Data Visualization, and Problem-Based Learning. He has published in 10 Science Citation Index (SCI) journals. He acted as a reviewer in many reputed journals, including, Future Generation of Computer Systems, Elsevier IEEE Transaction of Cloud Computing, IETE Journal of Research, IEEE Access, Journal of Grid Computing, The Journal of Supercomputing (JOS), and so on.

Dr. Tapas Badal is an Associate Professor in the School of Computer Science Engineering & Technology. He received B.Tech degree from the IES-IPS Academy, Indore, M.Tech from ABV-IITM Gwalior, and Ph.D. degrees from The MNIT Jaipur. He is having 10+ year of teaching and 6 years of research experience. He has published several good papers in reputed conferences and journals. His research interests include pattern recognition, computer vision and Natural Language Processing. He is interested in surveillance video activity analysis and knowledge extraction. He is currently involved in video-based studies of and efforts to support browsing and analysing activities. Also, he is working on AI-Bot and text analysis.

How to cite this article: Sahani Pooja Jaiprakash., H. Arya, I. Gupta, and T. Badal (2023), Energy Optimized Workflow Scheduling in IaaS Cloud: A Flower Pollination based Approach, , .