MAA: Multi-objective Artificial Algae Algorithm for Workflow Scheduling in Heterogeneous Fog-Cloud Environment

Prashant Shukla (pshukla.phd2019.it@nitrr.ac.in)
National Institute of Technology Raipur
Sudhakar Pandey
National Institute of Technology Raipur

Research Article

Keywords: Fog computing, Workflow Scheduling, Artificial Algae Algorithm, MAA Algorithm

Posted Date: December 2nd, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1871192/v1

Additional Declarations: No competing interests reported.

MAA: Multi-objective Artificial Algae Algorithm for Workflow Scheduling in Heterogeneous Fog-Cloud Environment

Prashant Shukla*, Sudhakar Pandey
Department of Information Technology,
National Institute of Technology, Raipur, Chhattisgarh, India, 492010.
E-mail: pshukla.phd2019.it@nitrr.ac.in*, spandey.it@nitrr.ac.in

Abstract— ***Cloud Computing (CC) is the most popular tool of choice for conducting scientific experimentation on Cloud Servers (CDs). It can be even more efficient strategy to use Fog Computing (FC) for allocating and executing operations on Fog Devices (FDs). Complex scientific operations need the effective use of virtual machines (VMs). Scientific workflow scheduling problem is regarded as NP-complete. This problem is constrained by various factors, such as Quality of Service (QoS), interdependence between tasks, user deadlines, etc. There is a very less research available on scientific workflow scheduling in Fog-Cloud Environments (FCE). Classical scheduling techniques, evolutionary optimization algorithms, and other methodologies are the available solution to this problem. In this paper, an efficient meta-heuristic approach named Multi-objective Artificial Algae (MAA) algorithm is presented for scheduling scientific workflows in heterogeneous FCE. In the first phase, the algorithm preprocesses scientific workflow and prepares a tasks list. In order to speed up execution, bottleneck tasks are executed with high priority. The MAA algorithm is used to schedule tasks in the following stage to reduce execution times, energy consumption and costs. In order to effectively use fog resources, the algorithm also utilizes the weighted sum based objective function. The suggested approach is evaluated using five benchmark scientific workflows. To verify the performance, the proposed algorithm's results are compared to those of conventional and specialized scheduling algorithms. In comparison to previous methodologies, the results demonstrate significant improvements in execution time, energy consumption and total cost without any trade-offs.***

Index Terms - Fog computing, Workflow Scheduling, Artificial Algae Algorithm, MAA Algorithm.

I. Introduction

A popular term in computer science for decades, cloud computing (CC) includes innovations including complexity abstraction and concealing, resource visualisation, and effective utilisation of distributed resources. GoGrid, Google App Engine, Microsoft Azure, and Amazon EC2 are a few popular CC platforms [1]. Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) are three categories into which CC services can be divided [2]. The calculation needs of real-time latency-sensible applications of highly dispersed Internet of things (IoT) systems are protected by a new computation pattern termed as "fog computing" (FC) [3]. In the cloud-to-things continuity, FC, reservations, management and networking are employed to distribute these services among end users [4]. The proof and development of the previous embedded systems is based on the FC platform that brings the cloud services onto the network edge [5]. We also see the challenge of IoT data processing as an interesting approach [6]. This system supports different applications including IoT, wireless fifth-generation (5G) and artificial intelligence embedded [4]. FC is good for reducing latency and cloud pricing, whereas CC is only useful to satisfy the rising demands of computer-intensive offloading programs [7]. Fog's primary characteristics are low latency and consciousness of the location, extensive geographical distribution, mobility, multiple nodes, a prominent role in availability of wireless, a substantial presence of flowing programmes and real time differences [8].

Workflows are modelled as Directed Acyclic Graphs (DAGs) with n tasks, where the vertices correspond to the tasks and the edges to their dependencies. Scientific workflows contain a huge number of tasks. These jobs also contain dependencies, which makes it challenging for the scheduler to organise tasks and use cloud resources effectively. The scheduler serves as a bridge between workflow tasks and cloud resources. Scheduling workflows in a cloud context is regarded as NP-complete. The efficiency of scheduling algorithms is affected by a variety of variables, including quality of service (QoS), user deadlines, financial cost, execution time, data privacy and security, etc. The vast computational resources required by workflow scheduling algorithms make them appropriate for FC systems. Workflow tasks can be carried out using scalable and affordable FC infrastructure. Workflow tasks vary in execution duration and computing demand [9]. While certain workflows require a lot of computing power, others may require a lot of memory and bandwidth.

Extensive literature study is available on workflow scheduling with CC settings while literature with FC settings is very rare. In order to overcome the issue, some academics have employed conventional scheduling algorithms, while others have concentrated on optimization techniques. There are single-objective, bi-objective, and multi-objective solutions. The majority of researchers have focused on makespan, cost, load balancing, etc. Heuristics or meta-heuristics are two possible approaches. The literature [11], [12] uses heuristics like min-min, max-min, etc. or combines these methods with meta-heuristics. In the Fog-Cloud Environment (FCE), a variety of meta-heuristic optimization methods have been employed to overcome workflow issues. To reduce the makespan, workflow scheduling employs the Genetic Algorithm (GA) [13]. Despite taking a little longer to get the solution, the GA method is reliable and produces a high-quality search in polynomial time. Particle Swarm Optimization (PSO) was employed by Pandey et al. [14] for scheduling workflow applications in a cloud computing environment. Although, PSO is a quick optimization technique, however it has drawbacks such early convergence and local optimal solution entrapment [15]. Recently, a meta-heuristic method called Grey Wolf Optimization (GWO) that imitates the leadership structure of grey wolves was suggested [16]. The enhanced version of Grey Wolf Optimizer was suggested by Khalil and Babamir [17] as a solution to the workflow issue.

The purpose of this research is to propose a meta heuristic algorithm for scheduling and distributing task across fog nodes in order to maximise the usage of network resources by users [6]. The following are our major contributions to this paper:

1) An efficient meta-heuristic approach namely Multi-objective Artificial Algae (MAA) Algorithm is proposed for workflow scheduling in Heterogeneous Fog Cloud Environment (HFCE). It includes a pre-processing stage to get tasks ready for the MAA algorithm. It is challenging for the scheduler to create an ideal plan since workflow tasks have interdependence. The dependencies are the focus of the proposed approach in order to achieve better scheduling.

2) The proposed workflow scheduling approach is evaluated on five realistic scientific workflow datasets like Montage, CyberShake, Epigenomics, LIGO, and SIPHT to optimize three performance objectives namely- total execution time (makespan), energy consumption and total cost.

The remaining part of the paper is structured as follows: The section II explains the classification of task scheduling strategies and section III examines the related work. In Section IV, we present our proposed task scheduling method. The simulation results are presented in Section V, and the conclusions are presented in Section VI.

II. Related Work

The concept of workflow, which divides a complicated scientific procedure into manageable activities, is quite well known among scientists [18]. These activities may be carried out via distributed and parallel computing, such as fog computing. In fog computing, workflow scheduling is a well-known NP-hard issue. In order to improve the efficiency of fog computing, a number of list-based algorithms have been proposed for task scheduling, including first come, first served (FCFS), round-robin (RR), shortest job first (SJF), minimal completion time (MCT), etc. The main principle of list-based heuristics is to prioritize each task and allocate the resources at hand in accordance with the preferences specified. In order to accommodate systems with heterogeneous multiprocessors, the Heterogeneous Earliest Finish Time (HEFT) was developed. When compared to current HEFT and Critical Path on a Processor (CPOP), Dubey et al [19] suggested improved version of HEFT can shorten the makespan.

The Min-Min method maps the job with the shortest possible execution time to the device with the shortest possible completion time [20]. A related technique is the Max-Min algorithm, which assigns the task with the longest possible execution time to the device with the shortest possible completion time. The tasks are not assigned to the resources as they enter when using the offline scheduling methods Min- Min and Max-Min, which operate in batch mode [21]. The problem with Min-Min and Max-Min algorithms is that they suffer from starvation [22]. In addition, they solely take time into account when evaluating resources. The user viewpoints are the exclusive focus of list-based heuristics; resource quality factors are given less attention.

These aforementioned standard heuristics methods are straightforward, simple to use, and quick, but meta-heuristic techniques can provide a solution that is almost optimal for complicated issues like workflow scheduling and can further increase the quality of the solution [23]. Additionally, heuristic algorithms are problem-dependent approaches, but meta-heuristic methods are problem-independent strategies. Because they are straightforward and offer powerful searching capabilities in a short amount of time and money, meta-heuristic algorithms are frequently employed. For overcoming the workflow problem, many meta-heuristic techniques were suggested. Ant Colony Optimization (ACO), Particle Swarm Optimization, and Genetic Algorithm (GA) are a few examples of common algorithms (PSO).

A genetically based approach for solving the job scheduling problem was proposed by Dasgupta et al. [1]. In comparison to Round Robing (RR), First Come First Serve (FCFS), and the local search method Stochastic Hill Climbing (SHC), the test's results demonstrate greater performance in terms of makespan. It has been claimed that the GA algorithm takes a long time to arrive at the best solutions [24]. In order to reduce the makespan, Tawfeek et al. [25] applied the Ant Colony Optimization (ACO) task scheduling method. They discovered that ACO outperformed FCFS and RR. Although, ACO is a fairly complicated algorithm, and it takes some time to achieve the best results [26]. Task dependencies are not taken into consideration. One of the well-known meta-heuristic methods is particle swarm optimization (PSO). It converges quickly and is easy to implement. Despite its benefits, it is unable to escape the local optimum for complicated problems [27].

The ability to exploit and explore is a characteristic of meta-heuristic algorithms [28]. Exploitation indicates how effective the algorithm is in conducting local searches. Exploring indicates that the approach may be used to identify the first answer, which may be close to the overall optimal value. An effective meta-heuristic algorithm strikes a balance between exploitation and exploration potential. Despite having a great capacity for exploration, particle swarm optimization has a limited capacity for exploitation. According to Mirjalili et al. [16], the grey wolf optimizer has a good blend of exploitation and exploratory abilities.

For complicated issues like scientific process scheduling, a single meta-heuristic could not yield the best solution and instead become trapped in the local best solution. Choosing one or more meta-heuristic algorithms and combining them according to their strongest traits is a superior strategy. Hybrid algorithms have gained popularity during the past couple decades. Only existing algorithms that are either hybrids of PSO or GWO are discussed here. The GA-PSO algorithm, which Manasrah and Ali [29] devised, is a combination of the Genetic Algorithm and Particle Swarm Optimization. Comparing the hybrid GA-PSO method to GA, PSO, and other algorithms, the overall execution time is decreased. Another hybrid method, a combination of the PSO and gravity search algorithm (GSA), has been published in [30]. In terms of cost, this hybrid method outperforms several non-heuristics, PSO, and GSA algorithms. Bouzary and Frank [31] suggested a combination of the Grey Wolf Optimization (GWO) and Genetic Approach (GA), and they discovered that the proposed algorithm outperformed the GWO and GA in terms of cost. When compared to flower pollination with genetic algorithms, Khurana and Singh's [32] hybrid flower pollination algorithm with GWO gives more effective outcomes while taking less time and money.

Although the aforementioned hybrid algorithms have advantages, one can wonder why the proposed technique was chosen. The free lunch theory [33] holds the key to the solution. According to the free lunch theory, a single method is ineffective for handling all optimization issues. It might perform better for a specific optimization issue. However, it might not perform well for the other optimization problems. Optimization issues don't have a single, universal answer.

III. BACKGROUND

In this section we first formulate the problem then discuss the fitness function used during the designing of the proposed algorithm. The standard Artificial Algae Algorithm (AAA) employed in the proposed approach is also described in this section.

A. Problem Formulation

DAGs are used to represent workflows. Dependencies among tasks in a workflow are defined as G = (V, E), where V denotes the vertices and stands for the tasks in the workflow and E denotes the edges and stands for dependencies between tasks. Prior to beginning the execution of any child task, the parent task must be completed [34]. Workflow tasks include several attributes, including execution time, data to be provided or received, and parent-child task relationships. Tasks in the workflow may be computation intensive or data-intensive, and sometimes both.

Several Fog Nodes (FDs), Cloud Datacenters (CDs) and End Devices (EDs) make up the FCE model. Numerous physical machines make up each FD/CD. Resources for computation and storage make up the physical machines. Each resource has the capability for processing, storage, memory, and bandwidth. Resources are shown as Virtual Machines (VMs) in FCE. The bandwidth, computing power, and cost of storage per unit of time for VMs are all fixed. Any of the resources that are accessible can execute workflow tasks scheduled for them. The quantity and strength of each Processing Element (PE) are used to calculate the processing capability of VM.

B. Fitness Function

The fitness function described the desired outcomes to be optimized with the proposed scheduling method [35]. A fitness function can be made multi-objective in two ways: priori and posteriori [36]. Each associated aim is given a weight based on its importance in the priori method, resulting in a single-valued function, also known as fitness value. In contrast, the posteriori technique uncovers the collection of non-dominant options. To design the fitness function, we use the priori technique. Makespan (MS) and Total Cost (*TC*) are the components of the fitness function. The considered fitness function can be described mathematically using the equation (1).

$$f(M) = \alpha1 \times MS + \alpha2 \times TC \tag{1}$$

Where M denotes mapping of workflow's n tasks to the m available VMs in EDs, FDs, and CDs, MS is for total execution time (makespan), EC stands for energy consumption and TC stands for total cost (computation and communication). The weights allocated to each aim are α1 and α2. We use a weight of 0.5 to equate the values of α1 and α2. The following sub-sections provide a comprehensive description of total execution time (makespan), total energy consumption and total execution cost:

1) Total execution time (makespan): The total execution time (makespan) is the time it takes for tasks in a workflow to complete. To put it another way, makespan is the amount of time it takes to complete all of the tasks assigned to various virtual machines [27]. The workflow's makespan can be calculated mathematically using equation 2.

$$MS_W = max\{CTi \mid i = 1,2,\ldots m\} \tag{2}$$

where CTi is the task Ti is completion time in the workflow. The entire time spent completing the tasks is the completion time. When tasks are interdependent, the time spent waiting for previous tasks is taken into account. Equation 3 represents the completion time CTi.

$$CTi = \begin{cases} ETi,\ iff\ pred(Ti) = \emptyset \\ WKi + ETi,\ iff\ pred(Ti) \neq \emptyset \end{cases} \tag{3}$$

As indicated in equation 4, the waiting time of task Ti is equal to the total completion time of all its predecessor tasks.

$$WKi = \begin{cases} 0,\ iff\ pred(Ti) = \emptyset \\ max(CTi),\ iff\ pred(Ti) \neq \emptyset \end{cases} \tag{4}$$

$$ET_{i,j} = \frac{SZ_{Task}}{Num(PE_i)\times PE_{Unit}} \tag{5}$$

Equation 5 is used to compute the execution time of task Ti on virtual machine VM_j, where SZ_{Task} is the task's size in million instructions (MI), Num(PEj) is the number of cores assigned to the virtual machine VM_j, and PE Unit is the size of each core in MIPS.

2) Total Cost: Because FC is based on a pay-as-you-go billing structure [37], cost is an important goal to minimize. The majority of fog service providers charge for a fixed period of time based on the fog services used. Execution, connection, and storage costs are all included in the cost of FC. The total execution cost of a VM is the product of the VM's cost per unit interval and the time it takes to complete tasks on that VM. The total execution cost (TC) of workflow W is calculated using equation 6 [38].

$$TC_W = \sum_{i\in W, i=1}^{k} \frac{ET_{i,j}}{\tau} \times CO_j : j \in VM_j \tag{6}$$

where CO_j is the cost of a type-i VM instance in the CD/FD per unit time. τ is the amount of time that the user uses the resources. $ET_{i,j}$ is the time it takes for type-j VM instance to complete task T_i.

3) Energy Consumption: The energy consumption is taken from [10], which contains active energy components denoted by E_{active} and idle energy components denoted E_{idle}. The E_{active} is related to the energy used while performing a task, whereas the E_{idle}, is referred to the energy consumed by idle resources. The term "active energy" can be determined using

$$E_{active} = \sum_{i=1}^{n} \alpha f_i v_i^2 \,(FT_{t_i} - ST_{t_i}) \tag{7}$$

where α is the constant, f_i represents the frequency and v_i represent the supply voltage for the resource on which task *i* is being performed. When idle, the resource enters a sleep state with a low power supply and less relative frequency. As a result, [10] is used to calculate the energy consumed over this period:

$$E_{idle} = \sum_{j=1}^{m} \sum idle_{jk} \in IDLE_{jk}\, \alpha f_{min_i} v_{min_i}^2 L_{jk} \tag{8}$$

where $IDLE_{jk}$ is a set of all idle slots of resource j. f_{mini} and v_{mini} represent the lowest supply voltage and frequency of resource j, respectively. L_{jk} is the amount of idle time for $idle_{jk}$. During the execution of tasks in the workflow, the overall energy consumed by the FCE is

$$EC = E_{active} + E_{idle} \tag{9}$$

C. Artificial Algae Algorithm

Using idealized versions of the attributes of algae, artificial algae are matched to each solution in the problem space. Artificial algae are similar to actual algae in that they can migrate toward the light source to photosynthesize by helical swimming, and they can adapt to their environment, alter the dominant species, and reproduce through mitotic division. The algorithm thus consisted of three fundamental components, referred to as "Evolutionary Process," "Adaptation," and "Helical Movement." Algae represent the primary genera in the algorithm. Algal colonies made up the entire population here. A collection of living algae is referred to as an algal colony as represented in equation (10) and (11) [39]. One algal cell split into two new algal cells, which live next to one another. When these two are divided again, another four cells live next to one another, and so on. Algal colonies functions like a single cell, moves as a unit, and its cells are susceptible to death in unfavorable environmental conditions. The colony may be divided into smaller parts by an external force like a shear force or by unfavorable conditions, and as life continues, each divided portion develops into a new colony. The colony that exists at the optimum point is known as the colony of optimums and is made up of the best-performing algae cells.

$$Population\ of\ Algal\ Colony = \begin{bmatrix} x_1^1 & \cdots & x_1^D \\ \vdots & \ddots & \vdots \\ x_N^1 & \cdots & x_N^D \end{bmatrix} \tag{10}$$

$$i^{th}\ algal\ colony = [x_i^1, x_i^2, \dots, x_i^D] \tag{11}$$

where x_i^j is algal cell in jth dimension of ith algal colony.

- *Evolutionary process:* When given adequate nutrients and light, an algal colony may expand and replicate, producing two new algal cells in time t, which is analogous to an actual mitotic division. On the other hand, an algal colony that doesn't get enough light persists for a time before passing away. The Monod model, which is provided in [39], was used to calculate the growth kinetics of the algal colony. Here, μ is the specific growth rate, *max* is the maximum specific growth rate, *S* is the nutrient concentration, which is the fitness value ($f^t(x_i)$) at time t in the model, and *K* is the algal colony's substrate half saturation constant. *max* was taken to be 1, as the conservation of mass principle states that the maximum amount that can be converted to biomass should be equal to the maximum amount of substrate that can be eaten in a given amount of time. *K* was calculated as the algal colony's growth rate in time t under circumstances of half nutrients. In the Monod equation, the size of the ith algal colony at time *t+1* is determined by the following equation [39]:

$$G_i^{t+1} = \mu_i^t G_i^t \qquad i = 1,2,\dots N \tag{12}$$

where *N* is the total number of algal colonies in the system and G_i^t is the size of the ith algal colony in time t.

Algal colonies that offer good solutions (the most effective and economical) grow more as the amount of nutrients they receive increases. In the course of evolution, an algae cell from the largest colony gets duplicated for every algal cell that dies in the smallest colony as shown in equation (13), (14) and (15) [39].

$$biggest^t = \max G_i^t, \qquad i = 1,2,\dots.N \tag{13}$$
$$smallest^t = \min G_i^t, \qquad i = 1,2,\dots.N \tag{14}$$
$$smallest_m^t = biggest_m^t, \qquad m = 1,2,\dots.N \tag{15}$$

where *biggest^t* represents the largest algal colony and *smallest^t* represents the smallest, and *D* represents the problem dimension. Algal colonies are arranged in AAA according to their sizes at time t. Algal cells from the smallest colony perish, whereas those from the largest colony multiply themselves in any randomly chosen dimension.

- *Adaptation:* When an algal colony struggles to expand adequately in a given environment, it tries to adapt, which changes the dominant species. An inadequately developed algal colony attempts to imitate the largest algal colony in its surroundings through the process of adaptation. The algorithmic modification to the starvation threshold puts an end to this process. For each artificial alga, the starvation value is initially set at zero. As algal cells receive inadequate light, starvation value rises over time t. The artificial alga with the highest starvation value according to equation (16) has evolved according to equation (17) [39]

$$starving^t = \max A_i^t, \qquad i = 1,2,\dots.N \tag{16}$$

$$starving^{t+1} = starving^{t} + (biggest^{t} - starving^{t}) \times rand \tag{17}$$

where A_i^t is the ith algal colony's starvation value at time t. $starving^t$ is the algal colony with the highest starvation value during time t. The application of the adaptation process at time t is determined by the adaptation parameter (A_p). A_p remains constant between [0, 1].

- *Helical movement:* Algal colonies and cells often swim and strive to remain near the water's surface because there, they can get enough light to survive. Thanks to the helically swimming motion of their flagella. Their flagella allow them to move ahead but are constrained by gravity and viscous drag as they swim helically in the fluid. Different algae cells move in different ways. Growing algal cells have a bigger friction surface, which enhances their capacity to conduct local searches and increases the frequency of helical motions. The amount of energy an algae cell has determines how much movement it can make. The quantity of nutrition taken in by an algal cell at time t is directly related to its energy level at that moment. As a result, an algae cell closer to the surface has more energy, giving it a better opportunity to travel around the liquid. On the other hand, because of the reduced friction surface, their moving distance in the liquid is greater. Consequently, they have a wider range of search possibilities. In contrast, they are less mobile in relation to its energy. An algae cell moves in a helical pattern, much like in real life. In AAA, viscous drag is represented as shear force, which is proportional to the size of the algal cell, and the gravitational force that restricts mobility is represented as 0. It has a spherical form, and the volume of it in the model determines its size. As a result, the friction surface is transformed into the hemisphere's surface area as represented in equation (18) and (19) [39].

$$\tau(x_i) = 2\pi r^2 \tag{18}$$

$$\tau(x_i) = 2\pi\left(\sqrt[3]{\frac{3G_i}{4\pi}}\right) \tag{19}$$

where (x_i) is the surface of friction. The three dimensions for the algal cell's helical movement are chosen at random. One of these allows for linear movement in Equation (20), and the other two dimensions allow for angular movement in Equations (21) and (22) [39]. Algal cells and colonies only migrate in one direction; hence equation (20) is utilized for one-dimensional issues. Since algal movement in two dimensions is sinusoidal, equations. (20) and (22) are used. Algal movement is helical when there are three dimensions or more, and equations (20)-(22) are employed. The movement's step size is determined by the friction surface and the distance from the light source:

$$x_{im}^{t+1} = x_{im}^{t} + \left(x_{jm}^{t} - x_{im}^{t}\right)\left(\Delta - \tau^{t}(x_i)\right)p \tag{20}$$
$$x_{ik}^{t+1} = x_{ik}^{t} + \left(x_{jk}^{t} - x_{ik}^{t}\right)\left(\Delta - \tau^{t}(x_i)\right)cos\alpha \tag{21}$$
$$x_{il}^{t+1} = x_{il}^{t} + \left(x_{jl}^{t} - x_{il}^{t}\right)\left(\Delta - \tau^{t}(x_i)\right)sin\beta \tag{22}$$

where x_{ik}^t, x_{il}^t, and x_{im}^t represent the ith algal cell's x, y, and z coordinates at time t, [0, 2]; p [1, 1]; Δ is shear force; and $\tau^t(x_i)$ represents the ith algal cell's friction surface area.

IV. PROPOSED ALGORITHM

This section starts with explaining how the tasks are scheduled for resources available and how this schedule can be utilized to build an optimal solution vector. It also introduces an efficient meta heuristic approach for scheduling workflows. Preprocessing the scientific workflows and AAA based scheduling optimization are the two main phases of the proposed MAA algorithm. Description of the proposed MAA algorithm is presented in the subsequent sections. Its flowchart is depicted in Figure 1 and 2, while the stepwise details are provided in Algorithms 1 and 2.

A. *Modeling the Solution Vector*

In this research, tasks can be scheduled to run on the resources like EDs, FDs, or CDs, as previously described. Concerning the sensor nodes, all computing resources have their processing capability and communication bandwidth. ED can only offload their tasks to FDs and CDs. They cannot offload their tasks to other EDs. As a result, only one representative ED is included in the encoding process for each sensor node when tasks are scheduled.

Each artificial algae cell of an algal colony is represented using natural numbers because task scheduling in FCE is a discrete problem. Like a solution is represented as individual chromosome in GA, particle in PSO, we used artificial algae cell in MAA. Here task-resource schedules are taken as artificial algae cells. Each vector has a length n equal to the total number of tasks in the workflow. Each index in the vector is a positive number representing the task number. The VM ID used to complete the task is the value supplied to this slot. The VM ID is chosen from all VMs accessible in the three-tier architecture of FCE. Assume a workflow includes ten scheduled tasks on five VMs: one ED, two FDs, and two CDs. The length of the individual, in this case, is ten, and each element is an integer between one and five. This individual's task assignment may look like this:

[4,3,2,4,5,4,2,1,5,1]. Table I and Table II show a more complete depiction of the solution vector and schedule.

TABLE 1: SOLUTION VECTOR EXAMPLE

Task ID ->	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
VM ID ->	4	3	2	1	5	4	2	1	5	1

TABLE II: TASK-RESOURCE SCHEDULING EXAMPLE

VM Layer	CD	FD	FD	ED	ED
VM ID	1	2	3	4	5
Assigned Task	4,8,10	3,7	2	1,6	5,9

B. *Fitness Function Evaluation*

The method starts with the computation of execution time and assigning these values to the makespan matrix, as depicted by equation (23). Each element value represents the execution time; for instance, $ET_{1,1}$ represents the execution time of task T_1 on VM_1. Using equation 5, the value of execution time in the matrix is computed.

$$MS = \begin{array}{c c} . & \begin{array}{ccc} VM_1 & \dots & VM_m \end{array} \\ \begin{array}{c} T_1 \\ \vdots \\ T_n \end{array} & \begin{bmatrix} ET_{1,1} & \cdots & ET_{1,m} \\ \vdots & \ddots & \vdots \\ ET_{n,1} & \cdots & ET_{n,m} \end{bmatrix} \end{array} \tag{23}$$

As stated in equation (24), a task dependency matrix (TD) can be used to depict the interdependence of tasks in a workflow. Each matrix entry is either 1 or 0. If $d_{1,2}$ equals 1, then task T_2 is performed after task T_1.

$$TD = \begin{array}{c c} . & \begin{array}{ccc} T_1 & \dots & T_n \end{array} \\ \begin{array}{c} T_1 \\ \vdots \\ T_n \end{array} & \begin{bmatrix} d_{1,1} & \cdots & d_{1,n} \\ \vdots & \ddots & \vdots \\ d_{n,1} & \cdots & d_{n,n} \end{bmatrix} \end{array} \tag{24}$$

As indicated in equation (25), the cost matrix records the execution cost per unit time for each VM. $C_1, C_2, \ldots, C_m$ represent the unit execution costs for VMs $VM_1, VM_2, \ldots, VM_m$ respectively.

$$TC = (C_1, C_2, \ldots., C_m) \tag{25}$$

As depicted in equation (26), the energy consumption matrix records the energy consumption per unit time for each VM. $EC_1, EC_2, \ldots, EC_m$ represent the unit energy consumption for VMs $VM_1, VM_2, \ldots, VM_m$ respectively.

$$EC = (EC_1, EC_2, \ldots., EC_m) \tag{26}$$

Based on above matrices, we can determine the makespan (*MS*), the energy consumption (EC), the total cost (*TC*), and the fitness function *f(M)* for each solution as described in equations (1)-(9) of Section III-B.

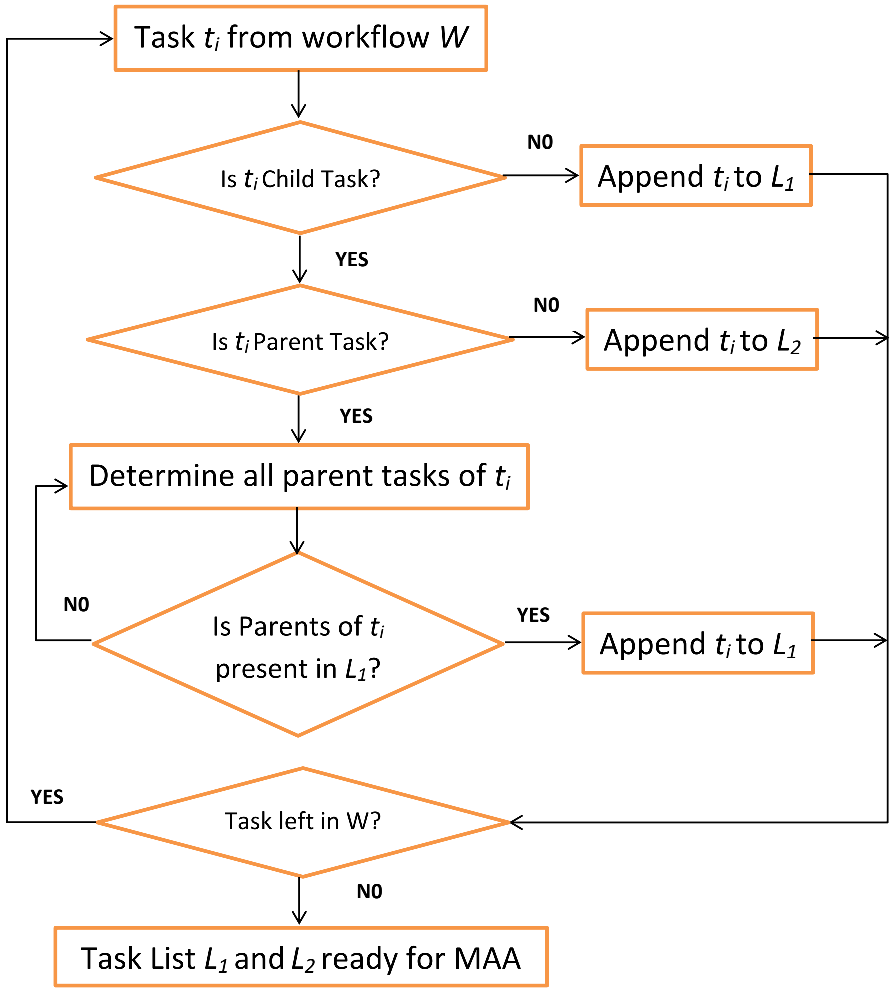
C. *Preprocessing the workflow*

The proposed approach employs preprocessing stages to prepare task lists and resource lists for MAA algorithm prior to its use. The suggested method organizes tasks based on the number of offspring; hence, tasks having a large number of descendants are handled first. These tasks act as a bottleneck for fog-based resources, resulting in lengthy execution durations [40]. The method also organizes fog resources based on their processing power, categorizing them as high-processing-power and low-processing-power resources. To perform workflow tasks, two resource lists are generated.

Parent tasks requiring a significant amount of execution time are executed on nodes with a high processing speed in order to swiftly reduce dependencies. After executing parent tasks, child tasks are executed based on their location in the graph, i.e. leaf tasks are executed with nodes with a low processing speed, while parent and intermediate tasks are executed with nodes with a high processing speed. Algorithm 1 uses lines 4-5 to separate the root tasks from workflow *W*. These root tasks are kept in a parent task list L_1. Now, the workflow is checked for leaf tasks in lines 6-7. These leaf tasks are transferred to child task list L_2. Before including the intermediate/dependent tasks in the list, line-8 first verifies the status of their parent tasks. If the parent task is already included in the list, then the intermediate tasks are transferred to parent task list else it has to wait till all its parent tasks are included. In line 16 two separate task-lists L_1 and L_2 are created for processing using MAA algorithm for workflow scheduling.

Algorithm 1: Preprocessing Phase of MAA algorithm for Workflow Scheduling

```
1.  Input   : workflow $W$
2.  Output  : task-lists $L_1$ and $L_2$
3.  for each task $t_i$ in $W$ do
4.          if $t_i$ is not a child task (root task) then
5.                  append $t_i$ in $L_1$
6.          else    if $t_i$ is not a parent task (leaf tasks) then
7.                          append $t_i$ in $L_2$
8.                  else    if all parent tasks of $t_i$ present in $L_1$ (intermediate tasks) then
9.                                  append $t_i$ in $L_1$
10.                         else
11.                                 wait till all parent tasks are appended in $L_1$
12.                         end-if
13.                 end-if
14.         end-if
15. end-for
16. task-lists $L_1$ and $L_2$ ready for MAA algorithm
```



Task t_i from workflow W
Is t_i Child Task?
NO
Append t_i to L_1
YES
Is t_i Parent Task?
NO
Append t_i to L_2
YES
Determine all parent tasks of t_i
NO
Is Parents of t_i present in L_1?
YES
Append t_i to L_1
YES
Task left in W?
NO
Task List L_1 and L_2 ready for MAA

Fig 1: Flow of the preprocessing phase of the proposed algorithm

Algorithm 2: Proposed Multi-objective Artificial Algae (MAA) Algorithm for Workflow Scheduling.

```
1.  Input: Task-lists $L_1$ and $L_2$ and Resource lists $R_1$ and $R_2$ from pre-processing phase
2.  Output: Optimal solution vector (Best mapping between tasks and resources)
3.  Define Algorithmic Parameters (shear force $\Delta$, energy loss $e$ and Adaptation parameter $A_p$)
4.  Initialize population size ($N$) and Maximum number of iterations ($MaxIter$).
5.  Randomly initialize $N$ algal colonies.
6.  Calculate colony size ($G_i$) for all algal colonies
7.  While ($t < MaxIter$)
8.          Evaluate energy ($E$) and Friction surface ($\tau$) of all algae
9.          For ($i$ = 1: $N$)
10.                 Starvation = true
11.                 While ($E(x_i) > 0$)
12.                         Select $j$ among all colonies via tournament selection method
13.                         Randomly select three dimensions ($k$, $l$ and $m$) for helical movement
14.                         Randomly generate angles $\alpha$ and $\beta$ in range $[0, 2\pi]$ and $p$ in range $[-1,1]$
                              $x_{ik}^{t+1} = x_{ik}^{t} + (x_{jk}^{t} - x_{ik}^{t})(\Delta - \tau^{t}(x_i))cos\alpha$
                              $x_{il}^{t+1} = x_{il}^{t} + (x_{jl}^{t} - x_{il}^{t})(\Delta - \tau^{t}(x_i))sin\beta$
                              $x_{im}^{t+1} = x_{im}^{t} + (x_{jm}^{t} - x_{im}^{t})(\Delta - \tau^{t}(x_i))p$
15.                         Evaluate new algal colony
16.                         Calculate energy loss due to movement, $E(x_i) = E(x_i) - (\frac{e}{2})$
17.                         If (new algal colony is better solution)
18.                                 Update current best colony/solution
19.                                 Starvation = false
20.                         Else
21.                                 Calculate energy loss due to metabolism, $E(x_i) = E(x_i) - (\frac{e}{2})$
22.                         End if
23.                 End While
24.                 If (Starvation = true)
25.                         Increment starvation, $A(x_i)$
26.                 End if
27.         End For
28.         Update colony size ($G_i$) for all colonies
29.         Randomly select one dimension for reproduction, $r$
30.         $smallest_r^t = biggest_r^t$
31.         If ($rand < A_p$)
32.                 $starving^{t+1} = starving^{t} + (biggest^{t} - starving^{t}) \times rand$
33.         End if
34.         Update current best colony/solution
35. End While
```

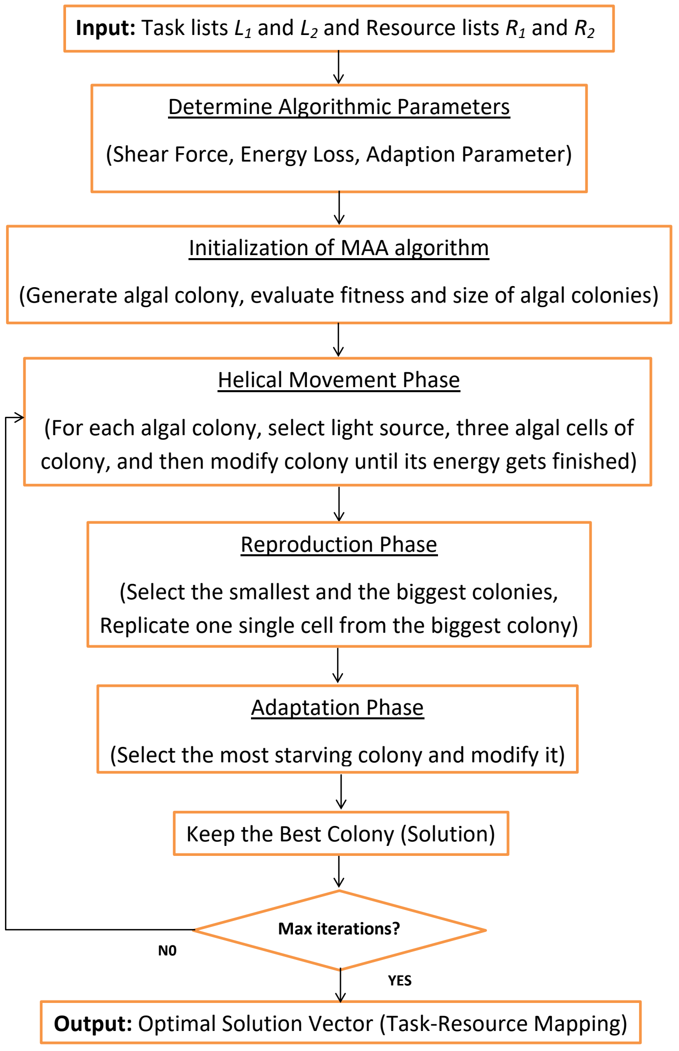

Input: Task lists L_1 and L_2 and Resource lists R_1 and R_2
Determine Algorithmic Parameters
(Shear Force, Energy Loss, Adaption Parameter)
Initialization of MAA algorithm
(Generate algal colony, evaluate fitness and size of algal colonies)
Helical Movement Phase
(For each algal colony, select light source, three algal cells of colony, and then modify colony until its energy gets finished)
Reproduction Phase
(Select the smallest and the biggest colonies, Replicate one single cell from the biggest colony)
Adaptation Phase
(Select the most starving colony and modify it)
Keep the Best Colony (Solution)
Max iterations?
NO
YES
Output: Optimal Solution Vector (Task-Resource Mapping)

Fig 2: Flow of Multi-Objective Artificial Algae (MAA) Algorithm for Workflow Scheduling

D. Applying the Multi-objective Artificial Algae (MAA) Algorithm

The proposed approach is based on AAA [39] technique to decrease workflow execution time (makespan) and cost while distributing the workload evenly across all computing nodes. Speedy and quicker algorithmic convergence are AAA's key advantages over other meta-heuristics. The technique has not been utilized in any literature on FCE for a variety of workflow tasks, including scheduling and resource allocation algorithms. It is expected that the scheduler is aware of how different workflow tasks are dependent on one another. Workflow task execution times are likewise predetermined. The proposed algorithm's goal is to allocate computing resources (VMs) to workflow tasks while minimizing their cost and execution time. The scheduling method aims to assign resource R_i to workflow task T_j in a way that makes effective use of computing resources. When allocating computing resources (VMs), the scheduler must take other factors into account. The MAA method assigns computing resources to tasks from both lists. The algorithm begins with randomly generating *N* algal colonies. Each colony representing a solution is assessed based on its fitness function value. Fitness value is determined by equation 1 based on the execution time and cost. During each iteration, all the variables are updated and the procedure is repeated until the halting requirement is not met.

It is quite difficult to design an effective mapping of tasks and resources [43]. We employ a search space with *n* dimensions for *n* tasks with a set of discrete potential values ranging from 1 to *m*, where *m* is the number of VMs. We employ notations from earlier research [44] to denote allocation of VMs to tasks i.e. $x_i^t = (x_{i1}^t, x_{i2}^t, \ldots, x_{ij}^t)$. Where x_{ij}^t indicates the VM_i is allocated to a jth algae cell of an algal colony at time t. The number of tasks in a workflow represents the dimension of an algal colony. The suggested algorithm stores viable solutions, i.e. colonies that are not dominated. The repository is initially empty. The repository is updated whenever the algorithm discovers a new solution. There are only non-dominated solutions in the repository. If the current solution is surpassed by any other solution during the process, the existing solution is replaced in the repository with the new solution. The judgement is based on the fitness criteria employed. In the final stage of the algorithm, the repository contains only viable solutions, which are non-dominated in nature.

V. PERFORMANCE EVALUATION

A. *Experimental Environment*

This section describes the experimental setup, followed by the findings and discussion of the experiment. The suggested technique was experimentally evaluated using scientific workflows [42] from various fields of study. Workflows are comprised of varying numbers of tasks, degrees of task dependencies, and data transmission between tasks. Some of the most practical scientific procedures, including Montage, CyberShake, Epigenomics, LIGO, and SIPHT, were published by the Pegasus project [45]. Figure 3 depicts the architectures of five scientific workflows. Table 3 provides information about some datasets utilized in tests. The algorithms were assessed based on their makespan, cost, and energy consumption. Makespan refers to the sum of all task execution times inside a workflow. Cost refers to the cost associated with the execution and transport of data for workflow application processes. Energy consumption is the matrix that indicates if the system's power consumption is optimal. Energy consumption is assessed as that of the summation of energy consumption during idle and active duty-cycles for all levels of computing resources. Minimum values are desirable for all performance parameters.

TABLE III: DATASETS USED IN THE EXPERIMENT

Dataset	Number of Tasks	Description
Montage	20,40,60,80,100,200,300	An astronomy tool developed by NASA/IPAC that generates unique sky mosaics from several input photos
Epigenomics	24, 47, 100	Utilized in bioinformatics to automate several activities in genome sequence processing.
SIPHT	30, 60, 100	Used to automate the search for untranslated RNAs (sRNAs) in the NCBI database for bacterial replicons.
LIGO	30, 50, 100	Used to create and analyze gravitational waves based on compact binary system coalescence data.
CyberShake	30, 50, 100	The Southern California Earthquake Center used it to quantify earthquake hazards in a given location.

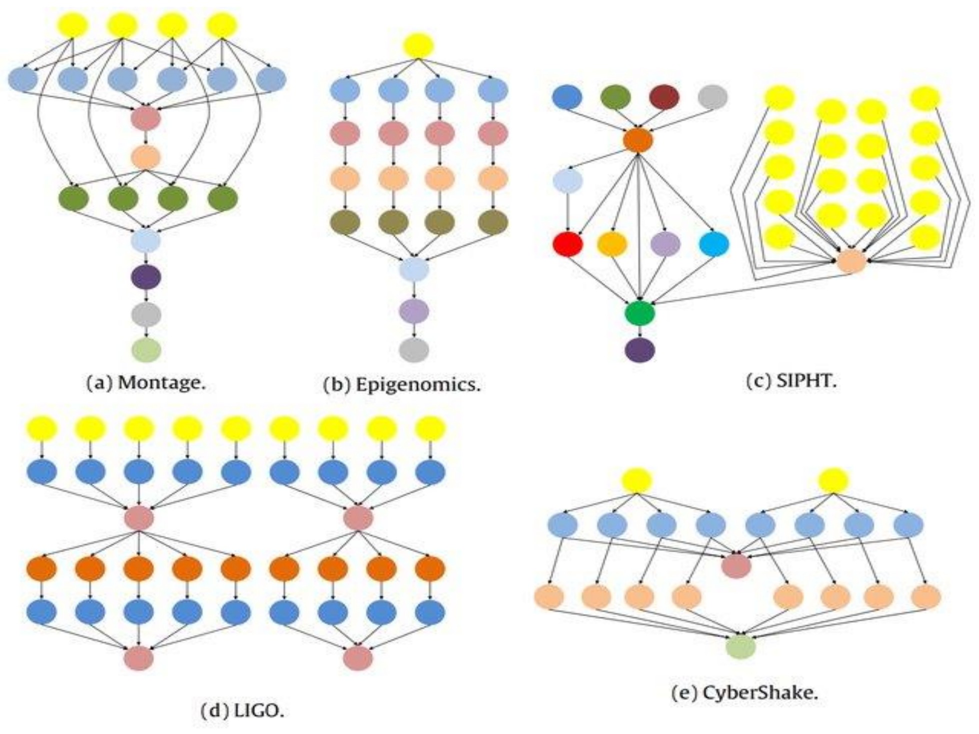


Fig 3: The structure of five realistic scientific workflows [42]

B. *Simulation Settings*

All the simulations are executed on a machine with a Windows 10 Pro 64-bit operating system, an Intel(R) Xeon(R) processor running at 3.70 GHz, and 16 GB of RAM. We used the Java IDE Eclipse to run the FogWorkflowSim-1.1 toolbox. FogWorkflowSim [40] is an extension of iFogsim that simulates user-defined task workflows in order to evaluate resource management strategies in FCE. The proposed approach is compared to Particle Swarm Optimization (PSO) [46], Ant Colony Optimization (ACO) [47], Grey Wolf Optimization (GWO) [48], and HPSOGWO [49] techniques. For each algorithm, the number of iterations and the size of population is taken as 100 and 25 respectively. The population size is assumed to be 25 for each algorithm. Table IV displays the simulation environment parameter settings for the three layers of HFCE. Table V displays the parameter settings used specifically for evaluating each method. We have modelled each technique for weighted sum objectives based on TIME, ENERGY, and COST. All three of the weighted coefficients w_1, w_2, and w_3 are set to the same value of 0.33. The algorithms can be evaluated using different realistic scientific workflows. The five scientific workflows that are under consideration are Montage, CyberShake, Epigenoimics, LIGO (Inspiral), and SIPHT. The Pegasus-generated scientific workflow structures are represented via a DAG XML file for each workflow [45]. These workflows are available with a range of task counts. For example, Montage is available with 20, 40, 60, 80, 100, 200, 300 and 1000 tasks. Simulations are conducted ten times for different combinations of workflow types and tasks counts in order to examine the algorithms' average performance.

TABLE IV. SIMULATION ENVIRONMENT SETTINGS

Parameters	**ED**	**FD**	**CD**
Number of Servers / Devices	5	5	5
Processing Speed (MIPS)	ED1-1000 ED2-1000 ED3-1000 ED4-1000 ED5-1000	FD1- 1200 FD2- 1300 FD3- 1400 FD4- 1500 FD5- 1600	CD1-1600 CD2-1700 CD3-1800 CD4-1900 CD5-2000
Task Execution Cost ($)	0	FD1- 0.1 FD2- 0.2 FD3- 0.3 FD4- 0.4 FD5- 0.5	CD1-0.5 CD2-0.6 CD3-0.7 CD4-0.8 CD5-0.9
Communication cost ($)	0	0.01	0.02
Active power (MW)	700	800	1600
Idle power (MW)	30	40	1300
Uplink bandwidth (Mbps)	20	10	1
Downlink bandwidth (Mbps)	40	10	10

TABLE V. ALGORITHM PARAMETERS OF WORKFLOW SCHEDULING.

Algorithm Parameters	**Values/Range**	**Explanation**
MaxIter	100	The total number of runs of the algorithm.
For PSO algorithm		
Particle Size	25	The number of particles, each particle represents a solution.
C_1, C_2	2	They are the acceleration coefficients.
W_p	0.1	It is inertia weight.
For ACO algorithm		
Ant count	25s	The number of ants, each ant represents a solution.
PR	0.1	They the Pheromone updating rate
CP	0.85	The choosing probability
W_a	0.95	The influence weights
For GWO algorithm		
Pack Size	25	The total number of wolves in a pack, each wolf is a potential solution.
a	[2,0]	This parameter decreases from 2 to 0 during the iterations of algorithm.
A	[-a, a]	This parameter is used to regulate the convergence rate, initially set to 0. when (A > 1) wolves diverge away from prey, when (A < 1) wolves converge towards prey.
C	[0,2]	This parameter is used to restrict falling into local optima, initially set to 0.
D	Any value	It is an empirical model used to surround the prey.
r_1, r_2	[0,1]	These are random coefficients. Their values vary from 0 to 1.
For MAA (Proposed)		
N	25	The number of algal colonies, each algal colony represents a solution.

e	0.3	The loss of energy
Δ	2	The shear force
A_p	0.5	The adaptation probability constant

C. *Results and Discussion*

The Result and Discussion has been divided into 3 parts based on the performance parameters evaluated during the experiment. In this section, we have discussed performance comparison of our suggested MAA algorithm with PSO [46], ACO [47], GWO [48], and HPSOGWO [49] algorithms. The performance is evaluated for five well-known workflows: Montage, CyberShake, Epigenomics, LIGO (Inspiral), and Sipht with task counts ranging from 20 to 300 in terms of Makespan (MS), Energy Consumption (EC), and Total Cost (TC). There was a limit of 100 iterations. Each scenario is performed 10 times before the average value of the result is taken into account. The simulation results are compiled in Tables VI, VII, and VIII. Fig. 3-17 shows the three-performance metrics concerning five workflows taking weighted sum based objective function.

TABLE VI. MAKESPAN FOR DIFFERENT WORKFLOW SETTINGS.

Scenario	**ACO**	**PSO**	**GWO**	**HPSOGWO**	**MAA**
Montage 20	0.29	0.22	0.25	0.21	**0.16**
Montage 40	**0.24**	0.32	0.29	0.34	0.30
Montage 60	0.36	0.50	0.37	0.38	**0.35**
Montage 80	0.46	0.44	0.41	0.47	**0.38**
Montage 100	0.61	**0.42**	0.45	0.62	0.47
Montage 200	0.95	0.95	**0.82**	0.90	0.83
Montage 300	1.54	1.24	1.48	1.51	**1.13**
CyberShake 30	70.43	68.51	74.44	64.40	**60.28**
CyberShake 50	90.50	83.93	83.90	95.63	**72.62**
CyberShake 100	100.79	93.43	101.60	102.13	**90.79**
Epigenomics 24	9.78	10.48	10.36	8.09	**6.96**
Epigenomics 47	14.16	14.89	16.31	14.20	**10.62**
Epigenomics 100	74.71	72.27	78.02	74.93	**57.90**
Inspiral 30	1.66	1.49	1.71	1.85	**1.08**
Inspiral 50	2.31	2.32	2.33	2.34	**1.51**
Inspiral 100	3.28	2.91	2.88	2.89	**2.34**
Sipht 30	3.92	3.74	3.56	4.06	**1.08**
Sipht 60	4.19	4.27	4.75	4.68	**1.51**
Sipht 100	5.36	4.65	5.06	4.75	**2.34**

TABLE VII. ENERGY CONSUMPTION FOR DIFFERENT WORKFLOW SETTINGS

Scenario	ACO	PSO	GWO	HPSOGWO	MAA
Montage 20	43	**26**	44	41	43
Montage 40	85	105	92	96	**83**
Montage 60	147	158	**148**	149	150
Montage 80	**202**	206	207	192	**202**
Montage 100	260	**243**	246	265	276
Montage 200	534	573	502	**497**	514
Montage 300	801	802	799	804	**774**
CyberShake 30	2332	6760	2290	2725	**2167**
CyberShake 50	3071	14771	2954	2878	**3178**
CyberShake 100	**3703**	24984	3739	3804	3706
Epigenomics 24	**4411**	3242	4634	4429	4647
Epigenomics 47	8490	**6719**	11599	10038	8800
Epigenomics 100	89090	80447	83415	97905	**77852**
Inspiral 30	1582	1463	1425	1544	**791**
Inspiral 50	2649	2838	2753	2507	**2400**
Inspiral 100	4906	4765	4706	5299	**4530**
Sipht 30	1700	791	**1105**	1339	1290
Sipht 60	**2205**	2400	3084	3491	2698
Sipht 100	4513	4530	5058	4542	**3664**

TABLE VIII. TOTAL COST FOR DIFFERENT WORKFLOW SETTINGS

Scenario	ACO	PSO	GWO	HPSOGWO	MAA
Montage 20	239	206	214	**161**	190
Montage 40	273	378	328	352	**329**
Montage 60	**424**	420	476	425	552
Montage 80	537	531	**471**	514	500
Montage 100	688	700	556	678	**533**
Montage 200	1002	987	**947**	1009	955
Montage 300	1589	1398	1530	1521	**1275**
CyberShake 30	65997	73673	52946	82856	**50756**
CyberShake 50	104854	115154	102463	112564	**90129**
CyberShake 100	182732	181543	171322	195005	**163653**
Epigenomics 24	8472	9813	8478	7563	**7021**
Epigenomics 47	14669	17121	**15087**	16510	15188
Epigenomics 100	120313	124962	118109	123156	**116871**
Inspiral 30	**1664**	2123	1686	1755	1746
Inspiral 50	2910	3038	**2825**	2970	3115

Inspiral 100	4941	5018	4788	5080	**4699**
Sipht 30	**1164**	2123	1416	1408	1264
Sipht 60	3084	3038	2662	2613	**2146**
Sipht 100	4029	5018	3968	3932	**3899**

The findings for makespan, cost, and energy usage for the Montage workflow are shown in Figures 3–5. As per expectations, all the measures increase with the number of tasks. The result demonstrates that ACO performs somewhat worse than all the other four techniques. On the other hand, out of the other three compared techniques PSO performs somewhat better. This is most likely due to PSO's most popular attribute of carrying out global search and local searches simultaneously. The MAA algorithm, which uses evolution and exploitation, clearly benefits from its capacity to see a wide variety of solutions thanks to the random reproduction and adaptation operators.

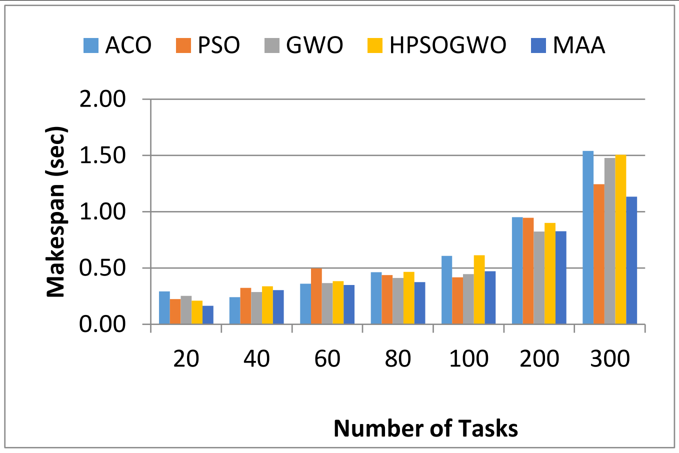


Fig 3: Makespan for Montage Workflow

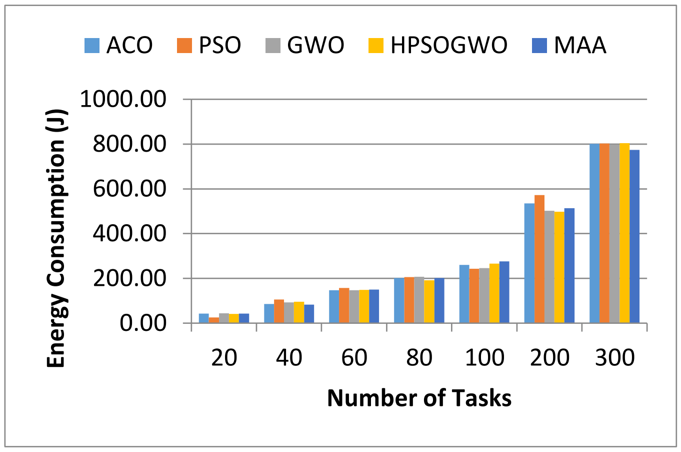


Fig 4: Energy Consumption for Montage Workflow

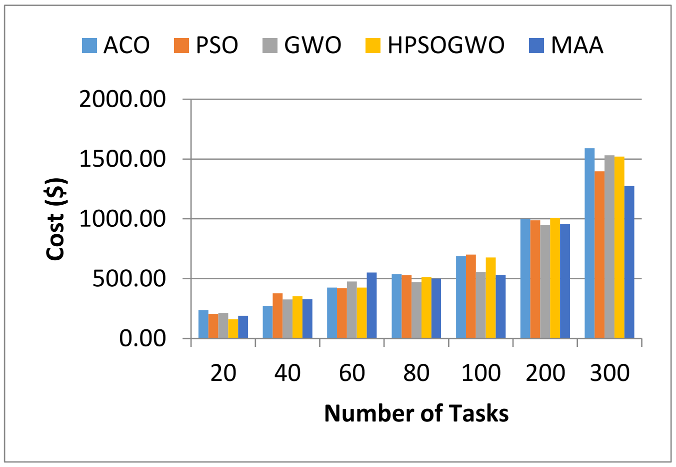


Fig 5: Cost for Montage Workflow

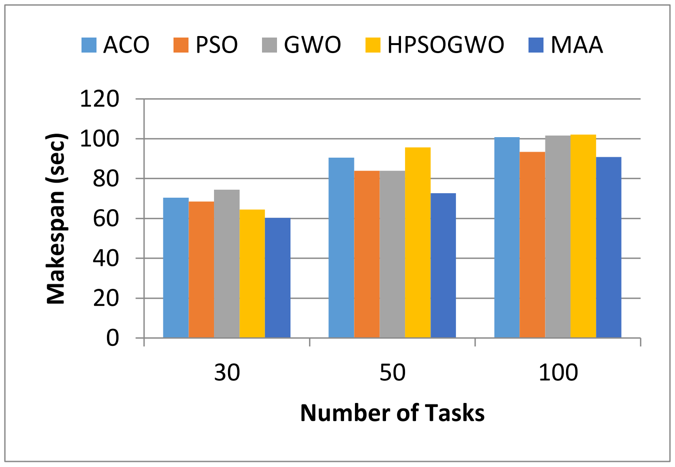


Fig 6: Makespan for Cybershake Workflow

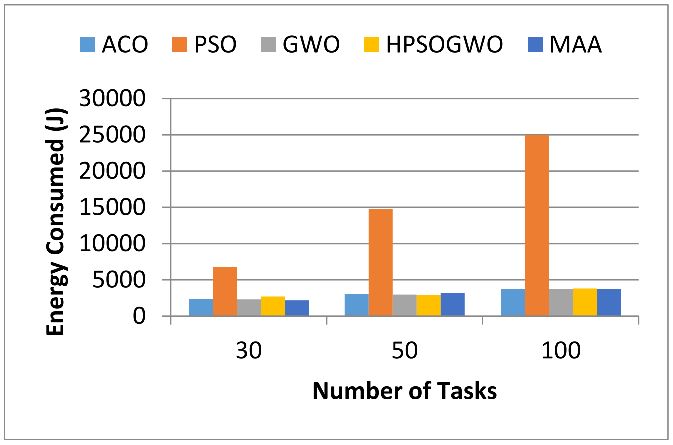


Fig 7: Energy Consumption for Cybershake Workflow

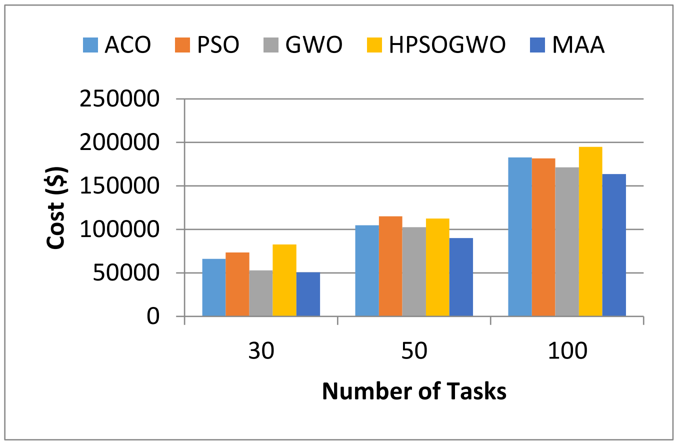


Fig 8: Cost for Cybershake Workflow

The findings for makespan, cost, and energy usage for the Cybershake workflow are shown in Figures 6-8. The results demonstrate that MAA performs better than the other four techniques. On the other hand, all the three techniques perform at a comparable level. However, there is an exception in the case of energy, since PSO performs worse than all the other alternatives in that regard. This is most likely due to PSO's prevalent issue with early convergence and becoming stuck in the local minima. The statistics for ACO, GWO, HPSOGWO and MAA tasks are pretty low, but PSO is high and rises rapidly as the number of tasks increases. For example, when makespan increases from less than 60 seconds for 30 tasks to more than 90 seconds for 100 tasks; energy rises drastically from less than 8KJ for 30 tasks to more than 25KJ for 100 tasks and price hikes from less than nearly eighty thousand dollars for 30 tasks to almost one lakh eighty thousand dollars for 100 tasks.

Figures 9–11 demonstrate the findings for the Epigenomics workflow in terms of makespan, cost, and energy usage. The statistics for 24 and 47 tasks are pretty small, but they rise dramatically as the number of tasks increases. For example, makespan increases from less than 20 seconds for 24 and 47 tasks to more than 60 seconds for 100 tasks; price hikes from less than ten thousand dollars for 24 and 47 tasks to more than one lakh dollars for 100 tasks; and energy increases from less than 10KJ for 24 and 47 tasks to more than 70 KJ for 100 tasks. This is because the mapping tasks in the Epigenomics workflow [23], which are responsible for matching genome sequences, become much more computationally expensive making turnaround time even longer as the number of activities increases. Compared to the other techniques, MAA still perform better in the case of makespan, energy and cost.

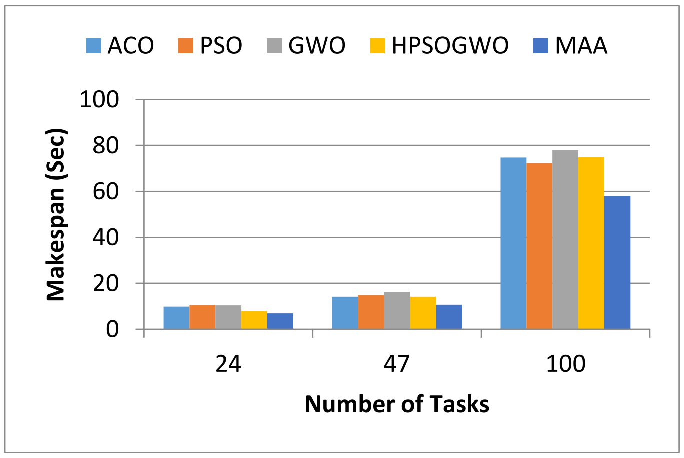


Fig 9: Makespan for Epigenomics Workflow

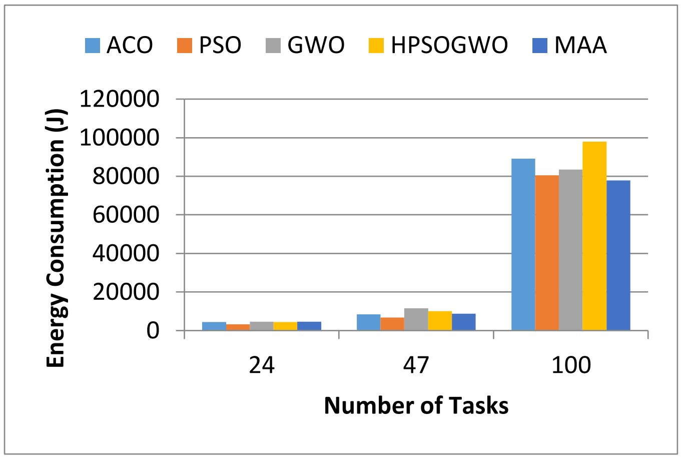


Fig 10: Energy Consumption for Epigenomics Workflow

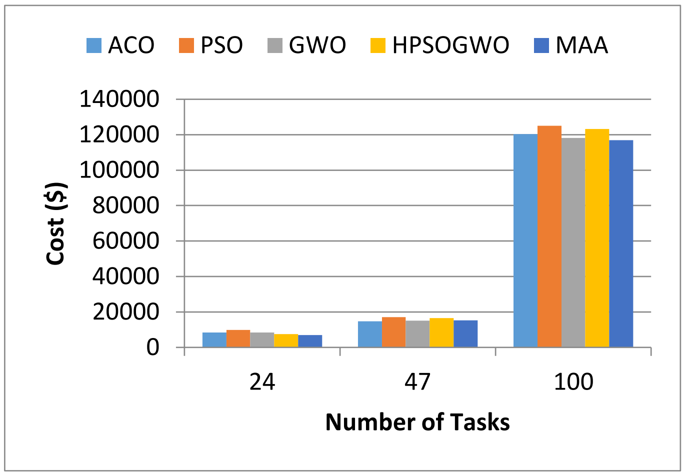


Fig 11: Cost for Epigenomics Workflow

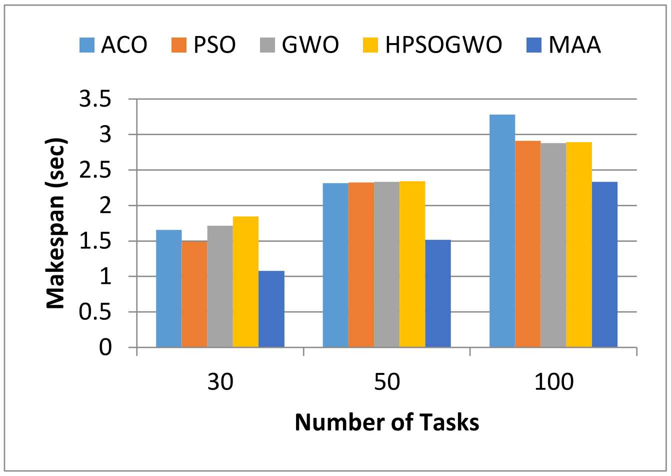


Fig 12: Makespan for LIGO Workflow

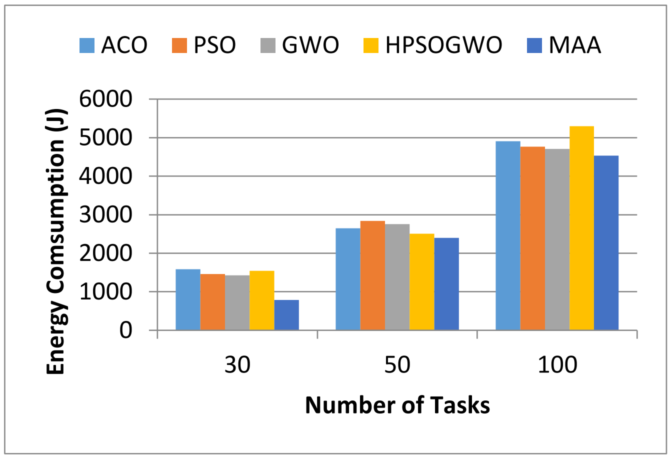


Fig 13: Energy Consumption for LIGO Workflow

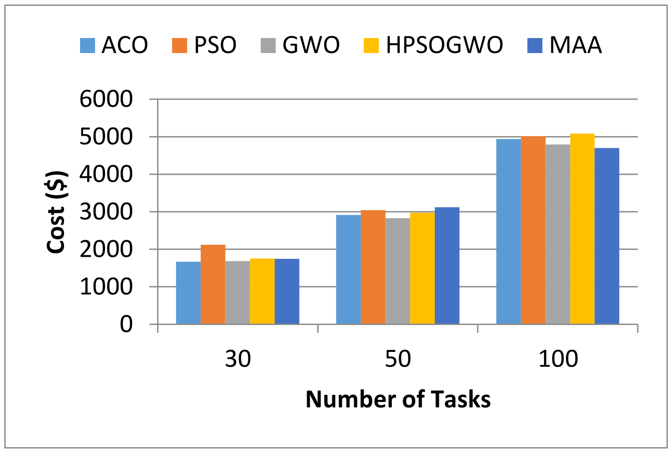


Fig 14: Cost for LIGO Workflow

The findings for makespan, cost, and energy usage for the LIGO workflow are shown in Figures 12–14. The makespan of all methods is consistently increasing in Fig. 12. MAA performs the best for all tasks, but ACO appears to perform worse than that of the prior three workflows. As shown in Fig. 13, with the rise in the number of tasks, the energy consumed by all algorithms are consistent except HPSOGWO which increases dramatically. As shown in fig. 14, with the increase in the number of tasks, all algorithms perform consistently. For 100 tasks, MAA continues to outperform all the other techniques. As the number of tasks rises, it appears that the HPSOWO algorithm distributes more tasks to FD/CDs, increasing the cost significantly. This results in significant energy usage on FCE, as demonstrated in Fig. 13.

The findings for makespan, cost, and energy usage for the SIPHT workflow are shown in Figures 15–17. The makespan of all methods is reasonably consistent in Fig. 15. MAA performs the best for all tasks, but ACO appears to perform worse than that of the other workflows. As the number of tasks rises, all algorithms except GWO perform consistently in Fig. 16. With the rise in the number of tasks, GWO increases dramatically than other techniques. As shown in Fig. 17, with the increase in the number of tasks, the cost of MAA continues to increase but is slower than the other techniques. Overall, it appears that with the rise in the number of tasks, the PSO algorithm distributes more work to FD/CDs, increasing the cost significantly. This results in significant energy usage on FCE, as demonstrated in Fig. 16.

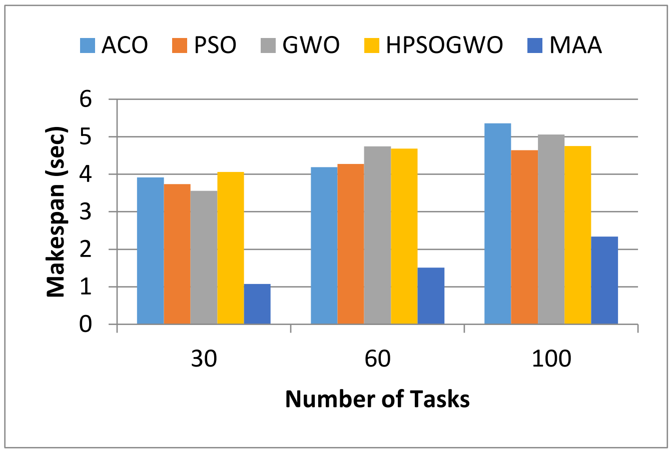


Fig 15: Makespan for SIPHT Workflow

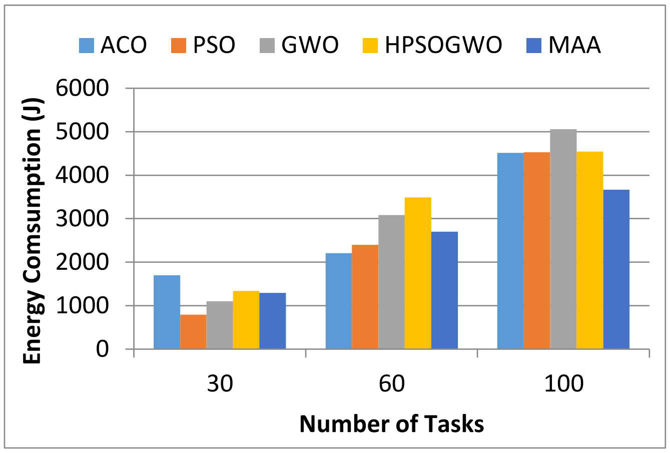


Fig 16: Energy Consumption for SIPHT Workflow

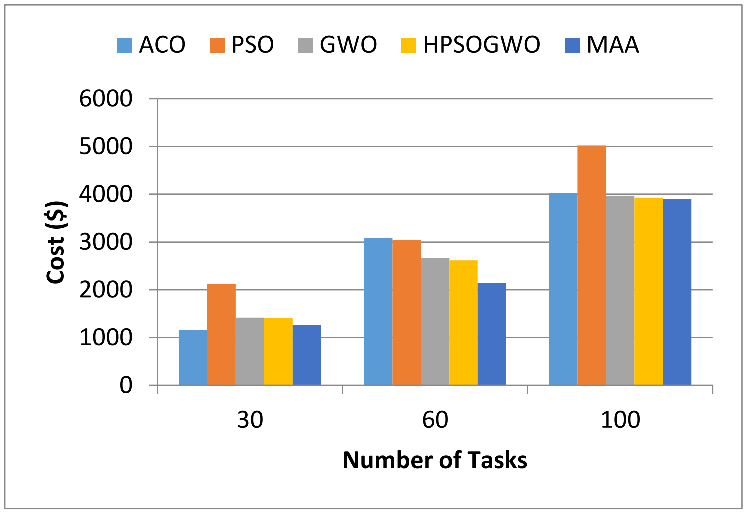


Fig 17: Cost for SIPHT Workflows

At first sight, the newly introduced algorithms seem to works properly, following logical thinking to reduce the makespan, energy consumption and cost. Although, some problems related to the objective specific scenarios can easily make it perform worse than the existing classic algorithms and provide the opposite desired result. By prioritizing the execution of the independent tasks to the available host/VM that guarantee the lowest makespan, energy and cost, we can provide incredible desired results. Here, scheduling a task to a VM may force another task to be later assigned on to another resource and may make the second task execution last a higher amount of time than necessary, which could finally increase the total execution time, power consumption, and cost as well.

VI. CONCLUSION

A Multi-objective Artificial Algae (MAA) algorithm for scheduling scientific workflows in heterogeneous FCE is presented in this article. The MAA algorithm targets to minimize a weighted sum objective function based on execution time, energy consumption and cost. At first, the proposed algorithm preprocesses the scientific workflows to remove bottleneck tasks. The algorithm is then used to schedule the preprocessed tasks list on the available VMs. The proposed approach is supported by experimental findings on scientific workflows taken from various research areas. With respect to the specified performance parameters like execution time, energy consumption and cost, the method outperforms all the existing algorithms.

This empirical study is bounded to 20-300 tasks to measure the performance of optimization algorithm for task scheduling in FCE. In future work, the number of tasks in workflows can be extended to 2000 to measure the effectiveness of the scheduling algorithms. Some more objectives like reliability, fault tolerance with the constraints like deadline and budget can also be considered. Considering some hybrid optimization approaches may also help to improve the overall performance of the FCE.

Declaration

Ethical Approval: Not Applicable.

Competing interests

The authors declare that they have no known competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

Authors' contributions:
Prashant Shukla: Conceptualization, Methodology, Implementation, Validation, Writing original draft
Sudhakar Pandey: Resources, Conceptualization, Writing-Review & Editing, Supervision.

Funding: Not Applicable.

Availability of data and materials: Not Applicable.

References

[1] K. Dasgupta, B. Mandal, P. Dutta, and S. Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing," Procedia Technology, vol. 10, pp. 340–347, jan 2013.
[2] A. Mazrekaj, I. Shabani, and B. Sejdiu, "Pricing Schemes in Cloud Computing: An Overview," International Journal of Advanced Computer Science and Applications, vol. 7, no. 2, pp. 80–86, 2016.
[3] Mahmud, R. and Buyya, R. (2016), "Fog computing: a taxonomy, survey and future directions", Internet of Everything, Springer, Singapore. arXiv preprint arXiv:1611.05539.
[4] Y. Zhao, Y. Li, I. Raicu, S. Lu, C. Lin, Y. Zhang, W. Tian, and R. Xue, "A service framework for scientific workflow management in the cloud," IEEE Trans. Services Computing, vol. 8, no. 6, pp. 930–940, 2015.
[5] W. Song, F. Chen, H. A. Jacobsen, X. Xia, C. Ye, and X. Ma, "Scientific workflow mining in clouds," IEEE Transactions on Parallel and Distributed Systems, 28(10), pp.2979-2992, 2017.
[6] M.A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments," Concurrency and Computation: Practice and Experience, 29(8), p.e4041, 2017.
[7] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," Parallel Computing, 62, pp.1-19., 2017.
[8] Y. Xie, Y. Wang, and Y. Yang, "A novel directional and non-local- convergent particle swarm optimization based workflow scheduling in cloud–edge environment," Future Gener. Comput. Syst., 97, 361–378, 2019.
[9] N. Anwar and H. Deng, "Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments," Future Internet, vol. 10, no. 1, p. 5, 2018.
[10] S. Yassa, R. Chelouah, H. Kadima, B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," Sci. World J. 2013, 2013.
[11] P. Kumar and A. Verma, "Scheduling using improved genetic algorithm in cloud computing for independent tasks," in Proceedings of the Inter- national Conference on Advances in Computing, Communications and Informatics. ACM, 2012, pp. 137–142.
[12] H. Xu, B. Yang, W. Qi, and E. Ahene, "A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery," KSII Transactions on Internet and Information Systems (TIIS), vol. 10, no. 3, pp. 976–995, 2016.
[13] J. Yu and R. Buyya, "A budget constrained scheduling of workflow applications on utility grids using genetic algorithms," 2006 Workshop on Workflows in Support of Large-Scale Science, WORKS '06, 2006.
[14] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in Proceedings - International Conference on Advanced Information Networking and Applications, AINA, 2010.
[15] M. Hosseini Shirvani, "A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems," Engineering Applications of Artificial Intelligence, vol. 90, no. September 2019, p. 103501, 2020.
[16] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, 2014.
[17] A. Khalili and S. M. Babamir, "Optimal scheduling workflows in cloud computing environment using Pareto-based Grey Wolf Optimizer," Concurrency Computation, vol. 29, no. 11, pp. 1–11, 2017.
[18] L. Zhang, L. Zhou, and A. Salah, "Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments," Information Sciences, vol. 531, pp. 31–46, 2020.
[19] K. Dubey, M. Kumar, and S. C. Sharma, "Modified HEFT Algorithm for Task Scheduling in Cloud Environment," Procedia Computer Science, vol. 125, pp. 725–732, 2018.
[20] G. Patel, R. Mehta, and U. Bhoi, "Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing," in Procedia Computer Science, 2015.
[21] R. Vijayalakshmi and V. Vasudevan, "Static batch mode heuristic algorithm for mapping independent tasks in computational grid," 2015.
[22] M. M. Golchi, S. Saraeian, and M. Heydari, "A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation," Computer Networks, vol. 162, p. 106860, 2019.
[23] K. L. Eng, A. Muhammed, M. A. Mohamed, and S. Hasan, "A hybrid heuristic of Variable Neighbourhood Descent and Great Deluge algorithm for efficient task scheduling in Grid computing," European Journal of Operational Research, 2019.

[24] Y. Ge and G. Wei, “GA-based task scheduler for the cloud computing systems,” Proceedings - 2010 International Conference on Web Information Systems and Mining, WISM 2010, vol. 2, pp. 181–186, 2010.

[25] M. Tawfeek, A. El-Sisi, A. Keshk, and F. Torkey, “Cloud task scheduling based on ant colony optimization,” International Arab Journal of Information Technology, vol. 12, no. 2, pp. 129–137, 2015.

[26] T. Deepa and D. Cheelu, “A Comparative Study of Static and Dynamic Computing,” 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), pp. 3375–3378, 2017.

[27] F. Ebadifard and S. M. Babamir, “A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment,” Concurrency Computation, vol. 30, no. 12, pp. 1–16, 2018.

[28] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bo¨lo¨ni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao, “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems,” Journal of Parallel and Distributed Computing, vol. 61, no. 4, pp. 810–837, 2001.

[29] A. M. Manasrah and H. B. Ali, “Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing,” Wireless Communications and Mobile Computing, vol. 2018, 2018.

[30] S. Mirzayi and V. Rafe, “A hybrid heuristic workflow scheduling algorithm for cloud computing environments,” Journal of Experimental and Theoretical Artificial Intelligence, 2015.

[31] H. Bouzary and F. Frank Chen, “A hybrid grey wolf optimizer algorithm with evolutionary operators for optimal QoS-aware service composition and optimal selection in cloud manufacturing,” International Journal of Advanced Manufacturing Technology, 2019.

[32] S. Khurana and R. Singh, “Workflow scheduling and reliability improvement by hybrid intelligence optimization approach with task ranking,” ICST Transactions on Scalable Information Systems, 2018.

[33] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization,” IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 67–82, 1997.

[34] L. Liu, M. Zhang, R. Buyya, and Q. Fan, “Deadline-constrained coevo- lutionary genetic algorithm for scientific workflow scheduling in cloud computing,” Concurrency and Computation: Practice and Experience, vol. 29, no. 5, p. e3942, 2017.

[35] F. Wu, Q. Wu, and Y. Tan, “Workflow scheduling in cloud: a survey,” Journal of Supercomputing, 2015.

[36] S. Mirjalili, S. Saremi, and S. Mohammad, “Multi-objective grey wolf optimizer : A novel algorithm for multi-criterion optimization,” Expert Systems With Applications, vol. 47, pp. 106–119, 2016.

[37] A. Pasdar, Y. C. Lee, and K. Almi’ani, “Hybrid scheduling for scientific workflows on hybrid clouds,” Computer Networks, vol. 181, no. August, p. 107438, 2020.

[38] M. Adhikari, T. Amgoth, and S. N. Srirama, “A survey on scheduling strategies for workflows in cloud environment and emerging trends,” ACM Computing Surveys, vol. 52, no. 4, 2019.

[39] Uymaz, S. A., Tezel, G., & Yel, E., “Artificial algae algorithm (AAA) for nonlinear global optimization” Applied soft computing, 31, 153-171., 2015

[40] K. Wu, “A tunable workflow scheduling algorithm based on particle swarm optimization for cloud computing,” 2014.

[41] X. Liu, L. Fan, J. Xu, X. Li, L. Gong, J. Grundy, and Y. Yang, “FogWorkflowSim: an automated simulation toolkit for workflow performance evaluation in fog computing,” In the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1114-1117), 2019.

[42] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.H. Su, and K. Vahi, “Characterization of scientific workflows,” In the Third Workshop on Workflows in Support of Large-Scale Science, pp. 1-10, 2008.

[43] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, “Cloudsim: a toolkit for modeling and simulation of cloud computing en- vironments and evaluation of resource provisioning algorithms,” Software: Practice and experience, vol. 41, no. 1, pp. 23–50, 2011.

[44] Sardaraz, M. and Tahir, M., “A hybrid algorithm for scheduling scientific workflows in cloud computing”, IEEE Access, 7, pp.186137-186146, 2019.

[45] “Pegasus,” [Online]. Available: https://pegasus.isi.edu/. [Accessed 20 August 2021].

[46] Subramoney, D., & Nyirenda, C. N. (2020, December). A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments. In 2020 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 760-767). IEEE.

[47] M. Tawfeek, A. El-Sisi, A. Keshk, and F. Torkey, “Cloud task scheduling based on ant colony optimization,” International Arab Journal of Information Technology, 2015.

[48] B. V. Natesha, N. Kumar Sharma, S. Domanal and R. M. Reddy Guddeti, "GWOTS: Grey Wolf Optimization Based Task Scheduling at the Green Cloud Data Center," *2018 14th International Conference on Semantics, Knowledge and Grids (SKG)*, 2018, pp. 181-187

[49] Arora, N., & Banyal, R. K., HPSOGWO: A Hybrid Algorithm for Scientific Workflow Scheduling in Cloud Computing. International Journal of Advanced Computer Science and Applications,11, no. 10 (2020).