A Hybrid Particle Whale Optimization Algorithm for Workflow Scheduling in Cloud-Fog-Mobile Computing Environment

Sumit Bansal (pup.sumit@gmail.com)
Punjabi University
Himanshu Aggarwal
Punjabi University

Research Article

Keywords: Scheduling, Workflow, Particle Swarm Optimization, Whale Optimization Algorithm

Posted Date: October 12th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2135828/v1

A Hybrid Particle Whale Optimization Algorithm for Workflow Scheduling in Cloud-Fog-Mobile Computing Environment

Sumit Bansal[1*], Dr. Himanshu Aggarwal[2]
[1,2]Punjabi University, Department of Computer Science & Engineering,
Patiala (Punjab), India
E-mail address: pup.sumit@gmail.com

Abstract – Cloud computing is the extensively used technology these days. Due to the usage of smart devices, a huge amount of data is produced. The processing of this data in real time is a big challenge for cloud servers. Fog computing is the solution for this, but fog has its own limitation in form of storage. To overcome, this cloud-fog architecture is preferred. In cloud-fog architecture, workflow scheduling is an open research area but finding an optimal algorithm is a major challenge. Some researchers proposed meta-heuristic algorithms to solve workflow scheduling issues but they are trapped locally and fails to give the global optimal solution.

To solve workflow scheduling problems, we propose the PWOA algorithm, a hybrid of Particle Swarm Optimization (PSO) and Whale Optimization Algorithm (WOA). The goal of this algorithm is to minimize the Total Execution Time (TET) and Total Execution Cost (TEC) of dependent tasks in a cloud-fog-mobile computing environment. Because it uses the features of both the standard PSO and WOA algorithms, the proposed algorithm overcomes the trapping problem also. In this article, the simulation results were compared to standard PSO and WOA algorithms using several benchmarks of four different scientific workflows (Cybershake, Epigenomics, Inspiral, Montage, and Sipht) with different numbers of tasks assigned in the proposed algorithm performed better.

Keywords: Scheduling, Workflow, Particle Swarm Optimization, Whale Optimization Algorithm

1. Introduction

Cloud computing is providing efficient use of distributed resources, abstraction of complexity, and virtualization features[1]. Nowadays, the use of smart devices is increasing drastically and huge amount of data is produced by these smart devices which is difficult for cloud servers to process in real time. To overcome this issue, CISCO has introduced new computing which is known as fog computing. It processes the data at the edges of the nodes, which elucidates the problematic issues of network bandwidth, utilization, latency, and storage. The use of only fog computing to solve the workflow scheduling problem is not enough because it doesn't have enough storage space and requires the help of cloud computing for it[2].

Scheduling is the mapping and planning of multiple tasks on shared resources. It assigns adequate resources to workflow tasks to ensure that the execution can be completed to meet different user-specified objective functions. Workflow scheduling is one of the key issues in the management of workflow execution and comes under the category of NP-Hard problem.

There are various types of scientific workflow tasks such as Montage, Sipht, Epigenomics, Cybershake and Inspiral, which are made available by the Pegasus project[3]. The Cybershake workflow is used to predict future geographical location by creating hazard maps and to analyse disaster modeling[4]. NASA employs the montage workflow in astrophysics to create unique mosaics of the sky from multi-input images[5] as well as to study and analyse the gravitational waveform. The spiral workflow is employed[6]. Epigenomics

workflow is a type of data processing pipeline. It is used for genome sequencing operations. Also, it is used at Epigenomic Center to generate DNA methylation and histone modification data[7]. Harvard maintains sipht workflow in applications for use in bioinformatics[8]. These are benchmark workflows and are used to compared with various scheduling algorithms.

The workflow scheduling problem lies in the category of mapping problems. The mapping of tasks with all the available virtual machines is a critical work. The usage of mapping is for the optimization of the objectives of the proposed scheduling problems. Figure 1 (below) shows a mapping of virtual machines (vm_1, vm_2.....vm_n) with tasks (t_1, t_2.....t_n). Here, the number of tasks and virtual machines are defined by the user.

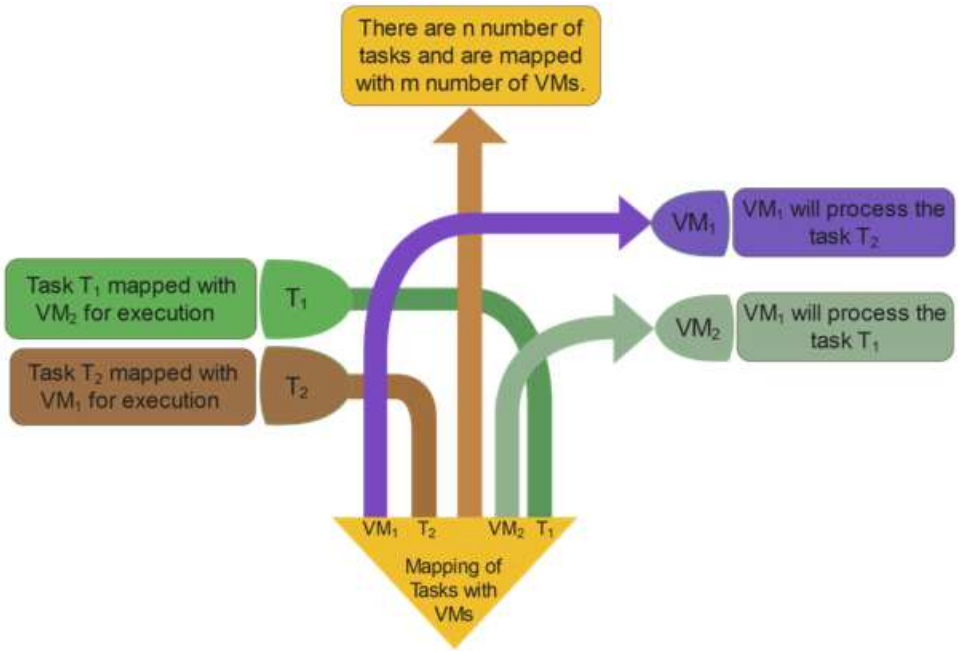

Figure 1: Mapping to Tasks with VMs

The workflow problems are solved with the use of meta-heuristic optimization algorithms in cloud-fog computing. The meta-heuristic algorithms are easy to implement and have strong searching power. It is more efficient in terms of time and cost. The most popular meta-heuristic PSO algorithm is used by S. Pandey et. al., [9]. They used the PSO algorithm to schedule the applications to cloud resources and achieve minimum computation and data transmission costs. PSO is a fast optimized efficient algorithm, and it has an earlier convergence problem and gets trapped in local optimal solutions. To overcome this problem, we have used the combination of PSO and WOA algorithm.

WOA algorithm mimics the social behavior of humpback whales. It reduces the probability of the occurrence of being trapped in the local optimal solution. Therefore, as result, a hybrid of the PSO-WOA algorithm is proposed. The performance of the proposed algorithm is evaluated on basis of five different workflows such as Montage, Sipht, Cybershake, Inspiral, and Epigenomics. The proposed algorithm's efficiency is measured in terms of TEC and TET when compared to standard algorithms. The simulation work is performed using the WorkflowSim simulation toolkit. The simulation results testifies that the proposed algorithm outperformed other algorithms.

1.1 Motivation and Comparison

In this field, a lot of research work has been discussed by the researchers but still, there are many unresolved issues related to workflow scheduling algorithms. A variety of research papers have been studied to highlight the issues such as parameters, hybridization, scheduling categories[10], workflow, resources types, environment, and simulation tools used. Researchers have implemented many hybrid meta-heuristic type algorithms to solve the above-mentioned issues but finding a universal solution to these problems is a challenging task. This challenge is the motivation to do this work to solve unresolved issues such as scheduling problems, hybridization, workflow, environment etc.. The comparison of major points of published papers in this field has been shown in Table 1 below.

Table 1: The comparative analysis between our work and previously published work

References	Year	Parameters		Hybridization	Workflow	Comparison with standard algorithm	Scheduling Environment			Scheduling Category	Resource Type	Simulation Tool
		Time	Cost				Mobile	Fog	Cloud			
Ahmed et al.[11]	2020	✗	✗	✓	✓	✗	✗	✓	✗	Heuristic	Hetero-genous	iFogSim
Subramoney and Nyirenda[12]	2020	✗	✓	✗	✓	✓	✗	✓	✓	Meta-Heuristic	Homo-genous	WorkflowSim
Liu et al.[13]	2021	✗	✓	✓	✗	✗	✗	✓	✓	Meta-Heuristic	Homo-genous	SUMO
Memari et al.[14]	2022	✓	✓	✓	✗	✗	✗	✓	✓	Meta-Heuristic	Hetero-genous	CloudSim
Rana et al.[15]	2021	✗	✓	✓	✗	✓	✗	✓	✗	Meta-Heuristic	Hetero-genous	CloudSim
Arora and Banyal et al.[16]	2021	✓	✓	✓	✓	✓	✗	✗	✓	Meta-Heuristic	Hetero-genous	WorkflowSim
Saoud and Recioui[17]	2022	✓	✗	✓	✗	✗	✗	✓	✓	Meta-Heuristic	Hetero-genous	Java Platform and Cloud Analyst
Thennarasu et al.[18]	2021	✓	✗	✗	✗	✗	✗	✗	✓	Meta-Heuristic	Hetero-genous	Hadoop under

												Linux environ-ment
Bisht and Vampugani[19]	2021	✗	✓	✗	✓	✗	✓	✓	✓	Heuristic	Hetero-genous	WorkflowSim
Arora and Banyal et al.[20]	2021	✓	✓	✓	✓	✓	✗	✗	✓	Meta-Heuristic	Hetero-genous	WorkflowSim
Alsmady et al.[21]	2019	✗	✓	✗	✓	✗	✗	✗	✓	Meta-heuristic	Hetero-genous	CloudSim
Ijaz et al.[22]	2021	✗	✗	✗	✓	✗	✗	✓	✓	Heuristic	Hetero-genous	MATLAB
Our Work	2022	✓	✓	✓	✓	✓	✓	✓	✓	Meta-Heuristic	Hetero-genous	WorkflowSim

Major key points of this research work are :

1. A meta-heuristic algorithm is implemented to solve the workflow scheduling problems.
2. The hybrid of PSO and WOA standard algorithms is discussed.
3. It helps in the reduction of local optimal problems in the proposed algorithm.
4. Scheduled the tasks on Cloud, Fog and Mobile computing environments.

The remainder of this paper is structured as follows. Section 2 explains the "Background" of the paper such as fitness function, standard PSO algorithm, gbest and pbest values, position and particle velocity, WOA algorithm, encircling prey, exploitation phase, and exploration phase of the paper. Section 3 named "Proposed PWOA algorithm" explains the approach to solve the formulated problem. Experimentation and comparison of results are provided in the "Performance Evaluation" section of the paper. Finally, the conclusion and the future works are presented in the "Conclusion and future works" section.

2. Background

Section 2.1 discusses the fitness function of the algorithm.

2.1 Fitness Function (FF)

The algorithm's parameters are optimized with the use of Fitness Function. This study evaluated two fitness functions, for a comparative assessment of the suggested approach. TEC is calculated with the use of the F1 fitness function and F2 is used to calculate TET respectively. Equations (1) and (5) define TEC and TET, respectively. The authors of this research article, minimized the TEC and TET specified by the Fitness Functions.

$$Fitness(F1) = Total\ execution\ cost\ (TEC) \text{.....(1)}$$

$$\text{TEC} = \sum VM\ C[i] * (LET[i] - LST[i]).\ i = 0 \ldots\ldots\ldots (2)$$

The computation of TEC is shown in Equation (2).

The i_{th} virtual machines TEC is the product of Least Start Time (LST), Least End Time (LET) and Cost (C[i]) of the virtual machines. C[i] is the i_{th} virtual machine's execution cost per unit time period. For all VMs, the value of C is assumed to be 1.

$$LET[i] = \max\ (ET[i]) \ldots\ldots\ldots\ldots\ldots (3)$$

$$LST[i] = \min\ (ET[i]) \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

The LET and LST are calculated by using equations (3) and (4), respectively. The maximum ET of the i_{th} VM is represented by LET of that VM. LST is the minimal ET of the i_{th} virtual machine.

The ET of the jobs performed on VM_0 and VM_1 is shown on the x-axis. VM_0 and VM_1 are shown on the y-axis. Tasks t_3, t_5, and t_6 are running on virtual machine VM_1. Virtual Machine (VM_0) is utilizing to complete tasks t_1, t_2, t_4, and t_7. LST (LST[1]) represent the beginning of t_3 on VM_1, whereas LET represents the end of t_6 on VM_1 (LET[1]).

The execution cost of VM_1 is calculated by multiplying the total difference between LET[1] and LST[1] with the execution cost (c[1]).TET is used to create another fitness function (F2), as shown in the equation below. As shown in equation (6) below, the TET is the maximum completion time of workflow task *Ti*.

$$Fitness(F2) = Total\ Execution\ Time(TET) \ldots\ldots\ldots\ldots (5)$$

$$TET_w = \max\ \{CT_i \mid i = 1,2,3,4 \ldots\ldots m\} \qquad \ldots\ldots\ldots\ldots (6)$$

In the workflow, CT_i is representing the completion time of job Ti's. TET is the completion time of the tasks. The waiting time of preceding jobs is also considered when tasks are dependent. The equation represents the completion time CT_i. (7).

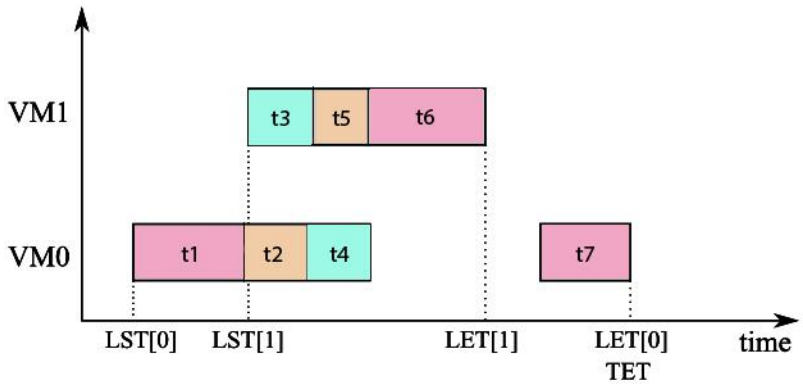

Figure 2: An example to calculate TET

$$CT_i = \begin{cases} ET_i & iff\ pred(T_i) = \emptyset \\ WK_i + ET_i & iff\ pred(T_i) \neq \emptyset \end{cases} \quad \text{.....(7)}$$

As shown in equation (8), the waiting time of assigned task Ti is the maximum completion time of the entire process predecessor tasks.

$$WK_i = \begin{cases} 0 & iff\ pred(T_i) = \emptyset \\ \max\,(CT_i) & iff\ pred(T_i) \neq \emptyset \end{cases}$$

$$ET_{i,j} = \frac{SZ_{Task}}{Num(PE_j) * PE_{Unit}} \quad \text{.....(8)}$$

Equation (8) is used to measures the ET of task *Ti* on virtual machine VMj, in which SZ_{Task} is the length of task *Ti* in million instructions, Num (PEj) is the total number of cores assigned to the virtual machine VM_j, and PE_{Unit} is the size for each and every core in MIPS. The computation of TET is shown in Figure 2 above. When task t_7 completes its execution, the workflow's TET occurs.

Section 2.2 discusses the standard PSO and WOA algorithms. It explains step-by-step interpretation and implementation of global and particle best values, particle velocity, next position, encircling prey, bubble net attacking method, etc. of algorithms respectively.

2.2 PSO Algorithm

Kennedy and Eberhart[23] introduced the PSO algorithm. PSO algorithm that follows the metaheuristic approach inspired by the social behaviour of the bird flock. Here each individual is known as a particle and total particles collectively are called population. Each particle represents a potential solution that is optimized over the jobs and VMs. As a typical metaheuristic, optimization has different stages discussed in sections 3.2.1 and 3.2.2 below.

2.2.1 Global best and Particle best values

In each iteration, the algorithm keeps track of each particle and traces the gbest and pbest values. Every solution is compared based on its fitness value calculated by the FF illustrated in section 2.1.

The solution having lowest fitness value ever achieved in all iterations performed is referred to as gbest value (global best). The best solution achieved by each particle ever is referred to as pbest value (particle best). Next, Algorithm 1 shows the pseudo-code to calculate the particle best (pbest) and global best (gbest) positions of particles.

Algorithm.1 Calculate pbest and gbest position of particles

Input: Particles
Output: pbest and gbest values
Set: pbest ← null; gbest ← null; i=0;
while i<totalparticles **do**
    if i=0 then
        pbest ← particle[i]
    end if
    if pbest.TEC>particle[i].TEC **then**
        pbest ← particle[i]
    end if
    if gbest.TEC>pbest[i].TEC then
        gbest ← particle[i]
    end if
end while

2.2.2 Update particles velocity and next position

With the course of iteration of the PSO algorithm, position and velocity are two factors that are updated. Particles move towards a more optimal position with each iteration based on velocity. The velocity of each particle is determined by the influence of the gbest value and its pbest value.

$$V_i(t+1) = w * V_i(t) + C_1.r_1 * \big(pbest - x_i(t)\big) + C_2.r_2 * \big(gbest - x_i(t)\big) \ldots\ldots\ldots (3)$$

The i_{th} particle's velocity at iteration t is represented in equation 3. The inertia weight, or w, is used to prevent an infinite increase in particle velocity. The particle's present position is supplied via $x_i(t)$.
The initial values for the coefficients C_1 and C_2 are shown in Table 2 below. The random integers between 0 and 1 are r_1 and r_2.

Table 2: Parameters used during implementation for PSO algorithm

Number of Particles	25
Number of Iterations	500
Inertia Weight	0.1
r_1, r_2	[0,1]
Learning Factor c_1	2
Learning Factor c_2	2
Repeated Experiment	15

Algorithm 2 and 3 shows the steps to update particle velocity and position.

Algorithm.2 Update velocity of particle

Input: Particle velocity value
Output: Updated particle velocity value

1 ***Set:*** *pbest ← null; gbest ← null; r_1,r_2 ← rand[0,1]; c_1,c_2 ← 2; w ← 0.1*
2 while i<totalparticle **do** **(i=index of particle)**
3 particle[i].vel[k] ← w * (particle[i].vel[k] + c_1 * r_1 * ((particle[i].pbestpos[k] – particle[i].pos[k])) + c_2 * r_2 * ((gbest.pos[k] – particle[i].pos[k]))))
4 end while

Algorithm.3 Update position of particle

Input: Updated particle velocity
Output: Updated particle position

1 ***Set:*** *pbest ← null; gbest ← null; gbest ← 0;*
2 while i<particleLength **do** **(i=index of particle)**
3 particle[i].pos[k] = particle[i].pos[k] + particle[i].vel[k];
4 end while

The simulation parameters that were taken during the evaluation of the suggested work are displayed in Table 3. During the simulation, this uses 2 cloud servers, 2 fog nodes, and 1 end device (mobile). In comparison to fog and end devices, cloud servers have less processing power and are cheaper. The amount of MIPS (shown below) has been applied across various servers to execute the tasks.

Table 3: Simulation parameters during the evaluation

No. of Tasks	20-1000		
No. of Virtual Machines	5		
Number of Cloud Servers	2		
Number of Fog Nodes	2		
Number of End Devices	1		
MIPS (Cloud / Fog / End Devices)	1600	1300	1000
No. of Processors	1		
RAM	512		
Virtual Machine policy	Time Shared / Space shared		

2.3 WOA Algorithm

To solve continuous optimization problems, a swarm-based intelligent algorithm is proposed. It mimics the hunting behavior of humpback whales called bubble new feeding behavior. WOA algorithm was proposed by Mirjalili and Lewis [24]. Each whale in this algorithm serves as a search agent and can be assumed as a potential solution. The superior performance of the algorithm shows that it is best compared to existing meta-heuristic algorithms. In the next sub-

sections, the mathematical models are derived based on some equations that are discussed below. *Table 4* below shows the initial parameter values of the WOA algorithm.

Table 4: WOA algorithm parameters

Search agents	25
Number of Iterations	500
Repeated Experiment	15
r_1, r_2	[0-1]
b	1

2.3.1 Encircling Prey

During the process, humpback whales encircle the prey. The best solution obtained so far in all iterations is treated as leader whale. Other search agents, whales are updating their position based on leader whale. This behavior is represented in equation (5) and equation (6)

$$\vec{D} = |\vec{C}.\overrightarrow{X^*}(t) - \vec{X}(t)| \qquad (5)$$

$$\vec{X}(t+1) = \overrightarrow{X^*}(t) - \vec{A}.\vec{D} \qquad (6)$$

The current iteration is represented by t, $\overrightarrow{X*}$ refers to the best solution ever achieved (leader whale), $\vec{X}$ is the position vector of the current search agent. $\vec{D}$ is used to model the process around the prey in surroundings mathematically. This value is determined by the best position and current search agent position. $\vec{A}$ and $\vec{C}$ is the coefficient vectors that can be calculated as follows in equation 7 and equation 8

$$\vec{A} = 2\vec{a}.\overrightarrow{r_1} - \vec{a} \qquad (7)$$

$$\vec{C} = 2\overrightarrow{r_2} \qquad (8)$$

The coefficient vector $\vec{A}$ is responsible for convergence $\vec{C}$ is used to avoid the trapping of search agents in the local optimum solution. Equation (12) is used to evaluate the linear value reduction from 2 to 0. The range of random variables r_1 and r_2 is between 0 to 1.

2.3.2 Bubble net attacking method (exploitation phase)

Two approaches are designed mathematically to observe and analyse the humpback whale bubble-net behaviour and shown in equation (9).

$$\vec{X}(t+1) = \begin{cases} \vec{X}^*(t) - A.D & if\ p < 0.5 \\ D'.e^{bl}.\cos(2\pi l) + \vec{X}^*(t) & if\ p \geq 0.5 \end{cases} \qquad \text{.....(9)}$$

Where p is a random number in the range [0,1]. If p < 0.5 then the position will be updated based on the distance between best and current search agent. For p ≥ 0.5, a spiral equation is introduced mentioned in the second part of the equation (9). A spiral equation is proposed to mimic the humpback whales helix-shaped movement. The constant b is used to define the shape of a logarithmic spiral. D' and l will be calculated using equations (10) and (11).

$$D' = |\vec{X}^*(t) - \vec{X}(t)| \quad (10)$$

$$l = (a2 - 1).r + 1 \quad (11)$$

r is the random variable between [0,1] whereas variable a2 decreases linearly from -1 to -2 with iterations according to equation (13).

2.3.3 Search for prey (exploration phase)

To provide exploration and exploitation, equation (12) decreases the controlling parameter from 2 to 0 over the course of each iteration, and Equation 13 decreases a2 from -1 to -2.

$$a = 2 * (1 - \frac{t}{N}) \quad (12)$$

$$a2 = -1 + t * \left(\frac{-1}{N}\right) \quad (13)$$

N is the total number of iterations and t is the current iteration.

The steps to determine the leader position i.e. best solution in whales is shown in Algorithm 4. Algorithms 5, 6 shows the process of attacking and haunting the prey respectively.

Algorithm.4 Calculate leader position

Input: Whales
Output: leader position
Set: leaderpos ← null
while i<totalparticles **do**
    if i=0 then
        leaderpos ← gbest
    end if
    if whales[i].TEC < leaderpos.TEC **then**
        leaderpos ← whales[i]
    end if
end while

Algorithm.5 Calculate A, C, a_1, a_2 and l values

```
Input: A, C, a1, a2 and l
Output: Updated A, C, a1, a2 and l
Set: r1 ← rand (0,1), r2 ← rand (0,1)
while i< maxiter do
        A ← 2 * a1 * r1 - a1
        C ← 2 * r2
        a1 ← 2 - i * (2 / maxiter)
        a2 ← 1 + i *((-1) / maxiter)
        l ← (a2 -1) * rand [0,1] + 1
        i = i+1
end while
```

Algorithm.6 Update whale position

```
Input: Whale position
Output: Updated whale position
Set: p ← rand (0,1
while i < whalelength do
        if p < 0.5 then
                if abs(A) >= 1 then
                        randleader ← populationsize * rand (0,1)
                        Xrand ← population[randleader]
                        D ← abs (C * Xrand[i] - X[i])
                        X[i] ← Xrand[i] - A * D
                end if
                else
                        D ← abs (C * leaderpos[i] - X[i])
                        X[i] ← leaderpos[i] - A * D
                end else
        end if
        else
                D = abs(leaderpos[i] - X[i])
                X[i] = (D * exp (1) * cos (l * 2 * math.PI) + leaderpos[i])
        end else
        i ← i+1
end while
```

3. Proposed PWOA Algorithm

This article has proposed a hybrid meta-heuristic algorithm called as PWOA algorithm. It is the combination of PSO and WOA algorithm. Section 3 compares the performance of the proposed algorithm (PWOA) to that of the standard PSO and WOA algorithms. Montage, Cybershake, Epigenomics, Sipht, and Inspiral are the five scenarios formed by five scientific

workflows. These workflows are tested with various task numbers, such as Montage, which is tested with 20, 60, 100, and 300 tasks.

The algorithms performance is measured by minimizing the TET and TEC by scheduling the workflow. In the first phase, some parameters have been initialized for the algorithm as shown in Table 5 (below). The proposed algorithm starts with a randomly generated population of particles, each particle representing a solution to the workflow issues. The populations random solution is implemented by executing half of the PSO algorithm's iterations $\frac{n}{2}$, where n represents the total number of iterations. The gbest outcome of $\frac{n}{2}$ iterations of PSO algorithm are utilized as a leader whale by WOA algorithm. Remaining half of the iterations $\frac{n}{2} + 1$ to n are executed with that initialized leader whale via WOA algorithm. Results generated by WOA algorithm are used as the concluding outcomes of proposed algorithm. The number of iterations[25] used in the proposed algorithm determines the accuracy of the results. Each iteration improved the outcomes and brought it closer to the optimal solution.

Following the evaluation of the proposed algorithm's results, we observed that performance of new algorithm is better as compared to standard algorithm. Because the PSO algorithm is generally trapped in an optimal local solution for complex and challenging NP-Hard problems like workflow scheduling, the suggested PWOA algorithm performs better than the PSO approach[26].

Table 5: PWOA proposed algorithm parameters

No. of particles	25
Particle's length	No. of tasks
No. of Iterations	500
Inertia weight	0.1
Repeated experiment	15
Learning factor C_1	2
Learning factor C_2	2

So, after saving the PSO algorithm outcomes, we fed those in to WOA algorithm. This confirms the excellent TEC and TET results achieved by the proposed PWOA algorithm. Proposed algorithm and its flow chart are shown in Algorithm 7 and Figure 3 (below) respectively.

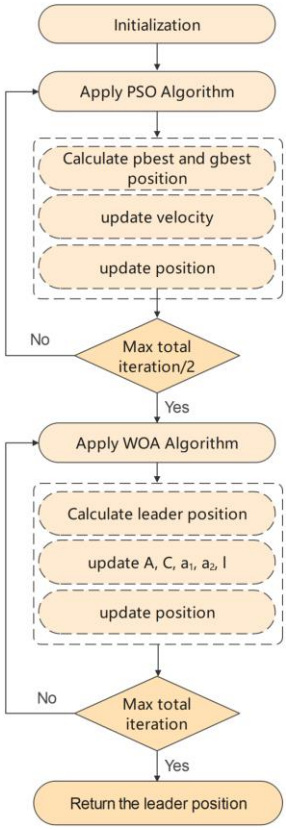

Figure 3: Proposed flow-chart of the PWOA algorithm

Algorithm.7 The proposed PSO-WOA algorithm

Input: Workflow W(N,E) and set of resource(VM_1, VM_2, ... VM_j)
Output: leaderpos

```
for i = 0 to populationsize do
    population ← randomize ()
end for
Set: i ← 0
while i < (maxiter/2) do
    calculate pbest and gbest position as per algorithm-1
    update velocity as per algorithm-2
    update position as per algorithm-3
    i ← i+1
  end while
Set: leaderpos ← gbest
  while I < maxiter do
    calculate leader position as per algorithm-4
    update A, C, a1, a2 and l as per algorithm-5
    update position as per algorthm-6
    i ← i+1
  end while
```

4. Performance Assessment

The PWOA algorithm is proposed to be implemented in all five scenarios to analyze the two Fitness Functions (F1 and F2) constituted for TET and TEC. Section 3.1 shows complete

details of fitness functions. To evaluate the results of PWOA algorithm are competed with WOA and PSO standard algorithms. The five given scenarios Cybershake, Epigenomics, Inspiral, Montage, and Sipht are having four scientific workflows each. All of these workflows are available with variable task numbers, such as Cybershake, which is available with 30, 50, 100, and 10000 tasks.

Table 6: The total average execution cost of various scenarios

Scenario	PSO	WOA	PWOA
Cybershake 30	230.89	810.71	171.85
Cybershake 50	578.67	2160.42	405.57
Cybershake 100	3927.26	10497.37	3247.67
Cybershake 1000	13270.35	24935.72	10932.23
Epigenomics 24	117.40	316.72	115.29
Epigenomics 47	190.73	536.01	192.78
Epigenomics 100	402.25	742.21	397.12
Epigenomics 997	3861.87	7961.67	3695.91
Inspiral 30	103.18	169.07	104.39
Inspiral 50	508.41	742.12	465.84
Inspiral 100	1304.74	1830.76	1202.39
Inspiral 1000	10937.11	15562.17	10715.21
Montage 20	8.43	13.355	8.55
Montage 60	42.25	70.09	41.76
Montage 100	56.63	84.39	56.14
Montage 300	136.37	198.31	134.109
Sipht 29	73.41	92.65	75.93
Sipht 58	190.09	259.98	176.81
Sipht 97	267.26	413.52	247.16
Sipht 968	3506.07	4530.24	3105.58

Tables 6 and 7 show the outcomes of three algorithms after 500 simulation runs in aspects of TEC and TET under various scenarios.

Table 7: Total average execution time of different scenarios

Scenario	PSO	WOA	PWOA
Cybershake 30	68609.04	97950.54	54665.45
Cybershake 50	88055.09	100024.52	95440.84
Cybershake 100	100873.41	110846.9	99786.92
Cybershake 1000	619587.85	693486.51	608563.73
Epigenomics 24	8744.954	16733.709	8043.32
Epigenomics 47	12213.77	27974.86	12538.36
Epigenomics 100	39691.36	71741.89	38441.45
Epigenomics 997	347462.57	609927.62	317426.81
Inspiral 30	1350.63	1672.026	1308.158
Inspiral 50	2066.46	4093.304	2072.64
Inspiral 100	3427.38	8270.407	3361.33
Inspiral 1000	31795.27	81359.64	29855.54

Montage 20	284.29	419.27	294.22
Montage 60	465.59	780.093	457.05
Montage 100	533.86	1351.84	503.32
Montage 300	1609.87	3637.34	1519.33
Sipht 29	3196.44	3623.57	3245.93
Sipht 58	6016.27	7410.23	5892.601
Sipht 97	9967.21	10992.203	9857.11
Sipht 968	98445.63	106482.68	97267.42

The proposed PWOA algorithm outperforms the standard PSO and WOA algorithms in terms of TEC and TET. Results of each scenario for different number of tasks are discussed in next section of the paper.

4.1 Experimental Results

The experimental outcomes of all five different scenarios such as Cybershake, Epigenomics, Inspiral, Montage, and Sipht is discussed here. These scenarios are available and each has a varying range of tasks, such as Cybershake, which has 30, 50, 100, and 1000 tasks. Experimental work is done with the help of a MacBook having a 1.8 GHz Dual-Core Intel Core i5 processor, 8 GB 1600 MHz DDR3 RAM, macOS Monterey Operating System version-12.3.1 (21E258). WorkflowSim, an extension of the CloudSim Simulation Toolkit, is used for implementation work.

4.1.1 Total Execution Cost:

Figure 4 depicts the TEC of various PSO, WOA, and PWOA algorithms for the Cybershake scenario at various task of numbers (30, 50, 100, and 1000). The TEC appears on the y-axis, and the number of tasks is shown along the x-axis. After evaluation of proposed algorithm, it is found that executing 1000, 100, 50, and 30 tasks the TEC is reduced by 17.62%, 17.30%, 29.91%, and 25.57% respectively in comparison to PSO. It is also seen that TEC decreased by 56.16%,69.06%, 81.22%, and 78.80% respectively concerning WOA.

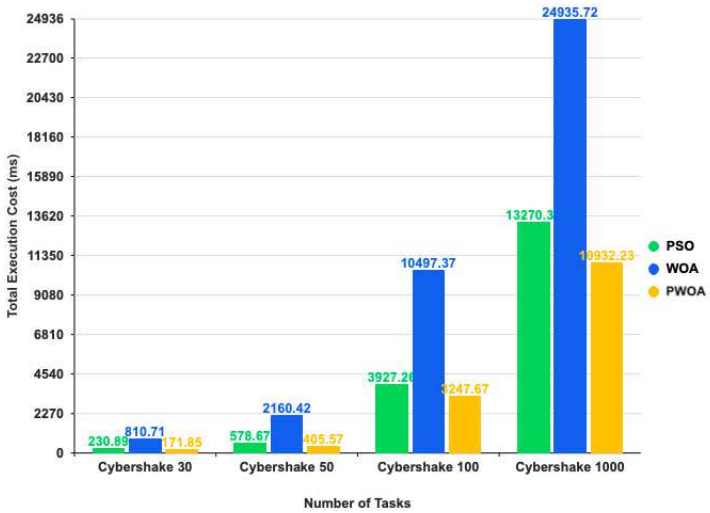

Figure 4: TEC in CyberShake workflow

For the Epigenomics scenario, a significant decrease in TEC is found during the evaluation of proposed algorithm in comparison with standard PSO and WOA algorithms. Evaluation results show that TEC decreased by 1.80%, 1.28%, and 4.30% for 24, 100, and 997 tasks in comparison to PSO. For 47 tasks TEC is increased slightly (1.07%) for PSO. While comparing with WOA, TEC decreased significantly for 24, 47, 100, and 997 tasks which are about 63.59%, 64.03%, 46.49%, and 53.57% respectively. The results of evaluation are shown in Figure 5 below.

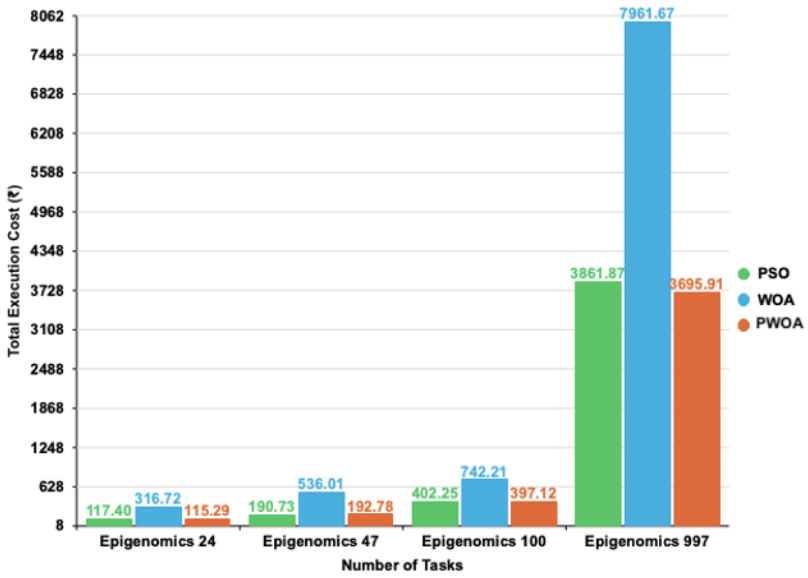

Figure 5: TEC in Epigenomics workflow

Figure 6 shown the Inspiral workflow comparison of TEC with PSO, WOA, and PWOA algorithms. In comparison to PSO, TEC in the new algorithm decreased by 8.37%, 7.84%, and 2.03% for 50, 100, and 1000 tasks, respectively, but increased slightly (1.17%) for 30 tasks. Similarly, when compared to WOA, the TEC is reduced by 38.25%, 37.22%, 34.32%, and 31.14% for 30, 50, 100, and 1000 tasks, respectively.

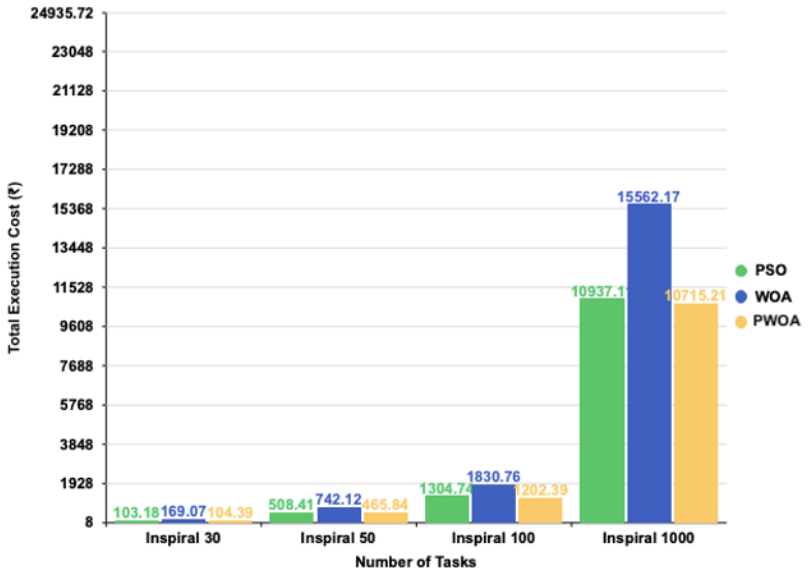

Figure 6: TEC in Inspiral workflow

Figure 7 below shows improvement in TEC results in the Montage workflow scenario. In the Montage workflow, the PWOA algorithm shows 1.16%, 0.87%, and 1.66% improvement respectively for 60, 100, and 300 tasks in comparison to PSO. For 20 tasks TEC is slightly increased (1.42%). Similarly, when the PWOA algorithm is compared with WOA results improved 35.97%, 40.41%, 33.47%, and 32.37% respectively for 20, 60, 100, and 300 tasks.

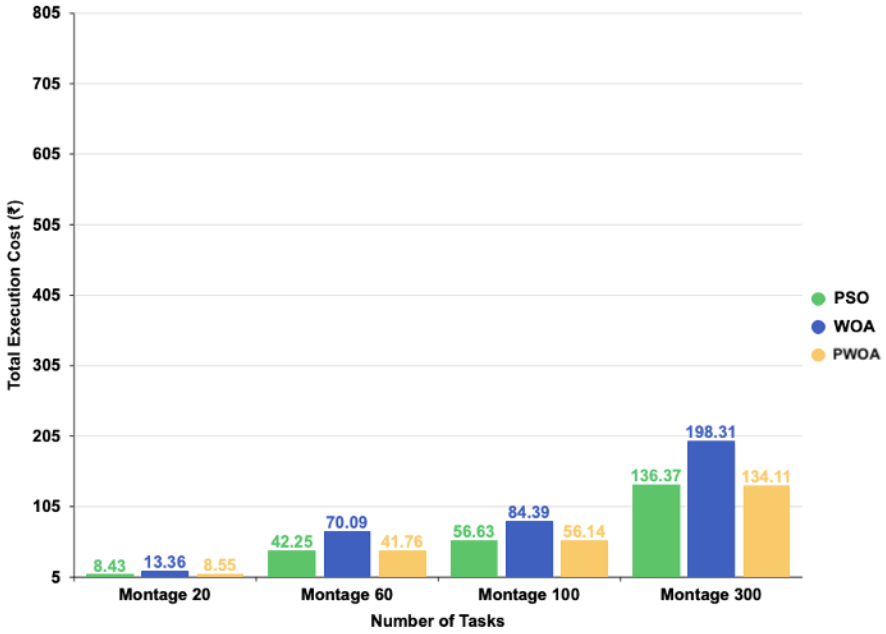

Figure 7: TEC in Montage workflow

As depicted in Figure 8 (below) TEC has improved in the Sipht workflow also. The results for 58, 97, and 968 tasks show that TEC is reduced by 6.99%, 7.52%, and 11.42% respectively when compared with the PSO algorithm. For 29 tasks TEC has some increase (3.43%). When it is compared with the WOA algorithm an improvement of 18.05%, 31.99%, 40.23%, and 31.45% has been observed respectively for 29, 58, 97, and 968 tasks.

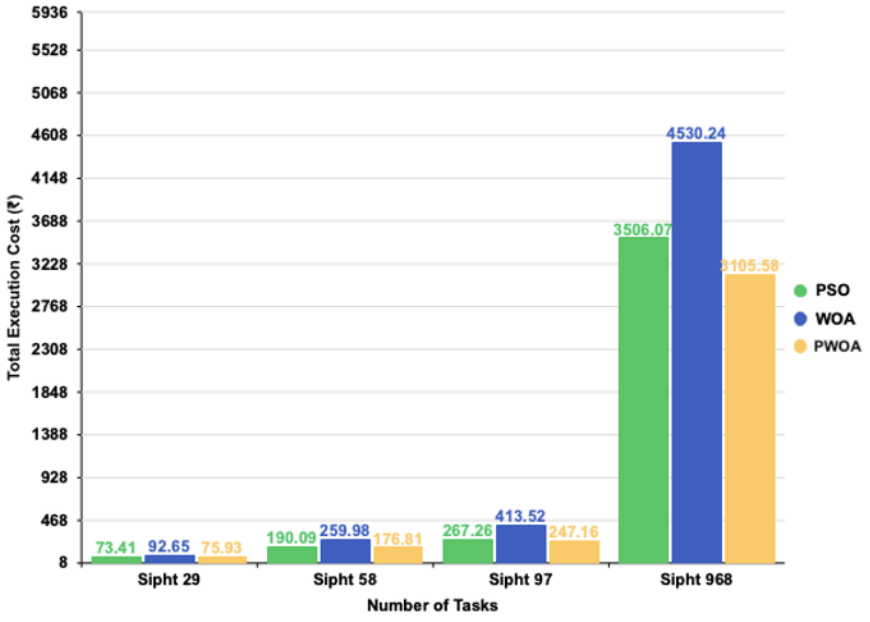

Figure 8: TEC in Sipht workflow

4.1.2 Makespan Time (TET):

The TET of different PSO, WOA, and PWOA algorithms for the Cybershake scenario at different number of tasks (30, 50, 100, 1000). Here, y-axis shows TET and x-axis shows the number of tasks. After assessment of proposed algorithm, it is found that while executing 30,100, and 1000 tasks, TET is reduced by 20.32%, 1.08%, and 1.78% respectively in comparison to PSO. Results show that for 50 tasks TET is slightly increased (8.39%). It is also seen that TET decreased by 44.19%,4.58%, 9.97%, and 12.24% respectively (for 30, 50, 100, and 1000 tasks) with respect to WOA algorithm as shown in Figure 9 below.

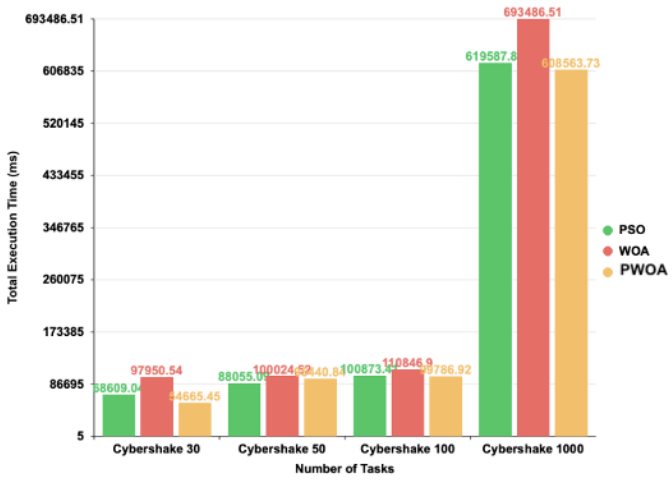

Figure 9: TET in Cybershake workflow

For the Epigenomics scenario, a significant decrease in TET is found during the execution of new algorithm as compared to standard PSO and WOA algorithms. Evaluation results show that TET is decreased by 8.02%, 3.14%, and 8.6% for 24, 100, and 997 tasks in comparison to PSO. For 47 tasks TET is increased slightly (2.66%) with respect to PSO. While compared with WOA, TET decreased significantly for 24, 47, 100, and 997 tasks which are about 51.93%, 55.17%, 46.41%, and 47.95% respectively. The results of the evaluation are shown in Figure 10 below.

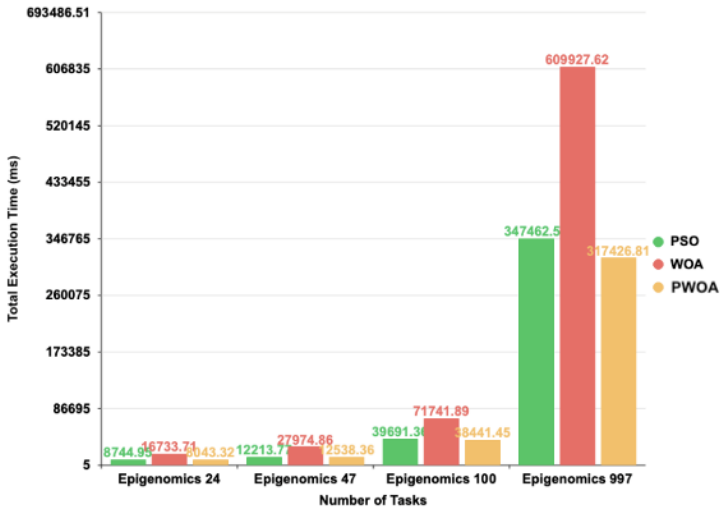

Figure 10: TET in Epigenomics workflow

Under Inspiral workflow comparison of TET of PSO, WOA, and PWOA algorithms is shown in Figure 11 below. TET decreased by 3.14%, 1.92%, and 6.10% for 30, 100, and 1000 tasks, respectively, when compared to PSO, but increased slightly (0.29%) for 50 tasks. Similarly, in comparison to WOA, the TET is reduced in 30 tasks by 21.76%, in 50 tasks by 49.36%, in 100 tasks by 59.36%, and in 1000 tasks by 63.30% respectively.

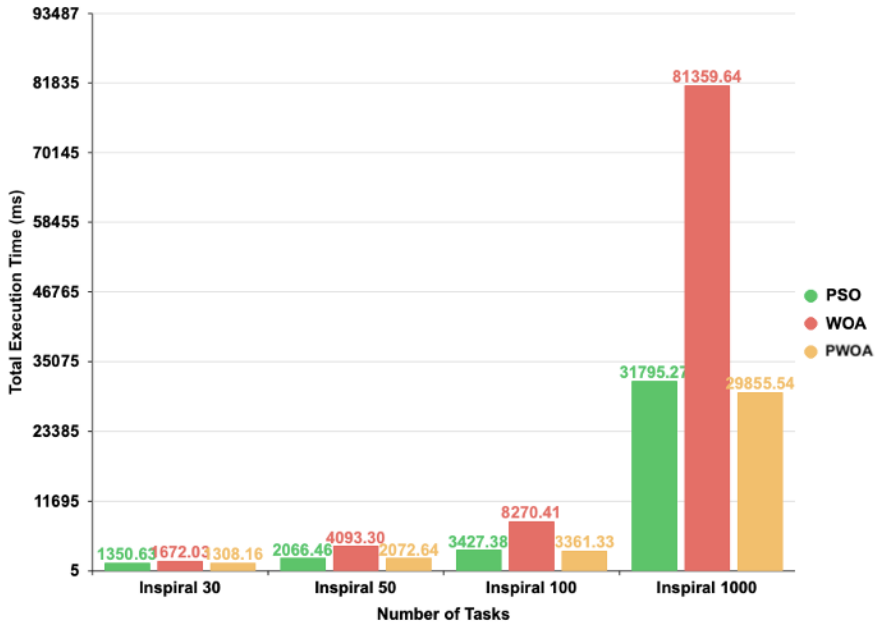

Figure 11: TET in Inspiral workflow

Figure 12 below shows improvement in TET results in the Montage workflow scenario. In the Montage workflow, PWOA algorithm shows 1.83%, 5.72%, and 5.62% improvement respectively for 60,100 and 300 tasks in comparison to PSO. For 20 tasks TET is slightly increased (3.49%). Similarly, when the PWOA algorithm is compared with WOA results improved by 29.82%, 41.41%, 62.76%, and 58.23% respectively for 20, 60, 100, and 300 tasks.

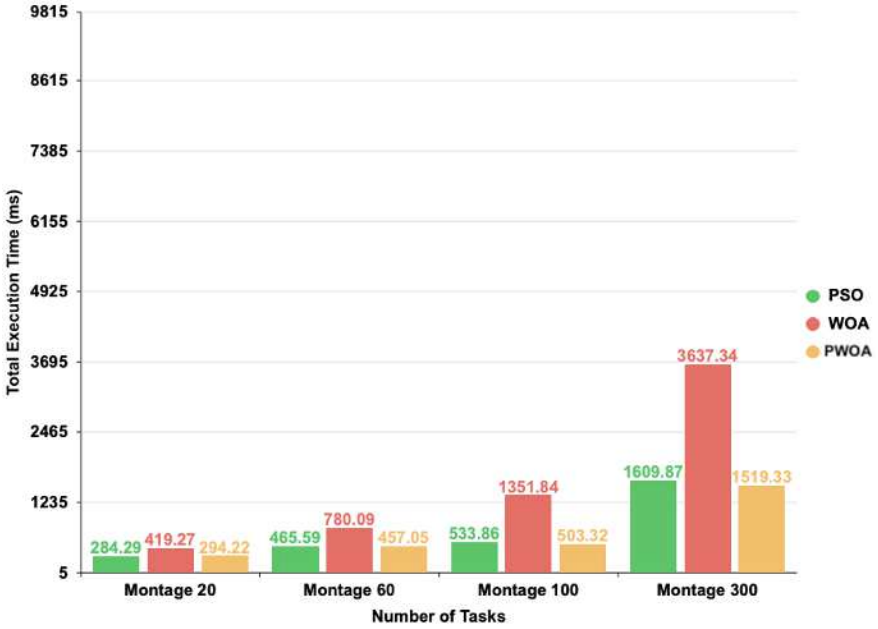

Figure 12: TET in Montage workflow

As depicted in Figure 13 below TET is improved in the Sipht workflow also. The results for 58, 97, and 968 tasks show that TET is reduced by 2.05%, 1.10%, and 1.20% respectively when compared with PSO algorithm. For 29 tasks, TET increased by (1.55%). When it is compared with WOA algorithm an improvement of 10.42%, 20.48%, 10.33%, and 8.65% has been observed respectively for 29, 58, 97, and 968 tasks.

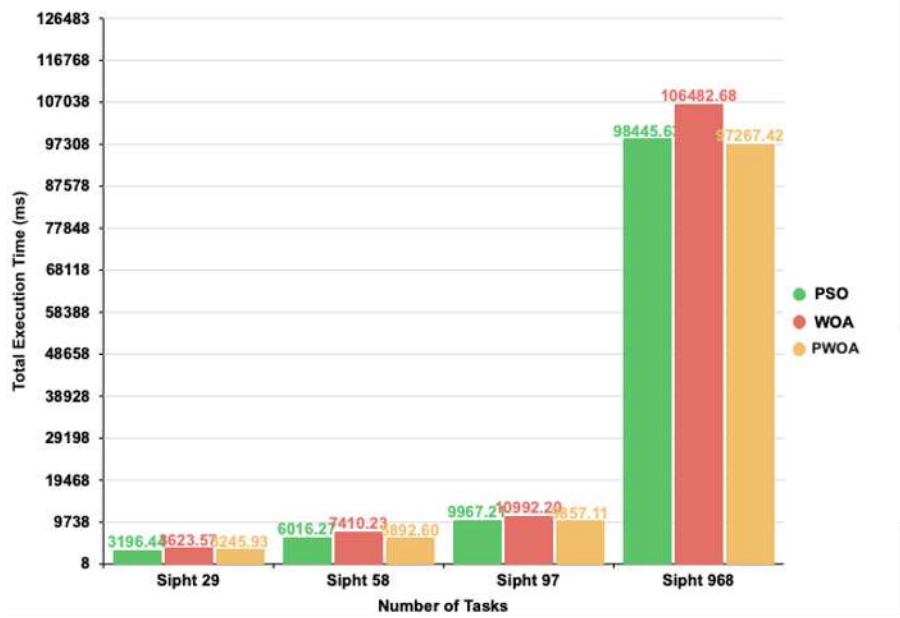

Figure 13: TET in Sipht workflow

4.3 Heterogeneous Environment Assessment

In real-time, system configuration of the cloud-fog system could be different so the test results can vary. In this manuscript proposed, algorithm tests have been done in a heterogeneous environment. The length of the tasks can vary according to the different number of tasks. For actual results, tasks length is required to be same. So, in heterogeneous systems, results can be variable. We have tested the results for different scenarios under 30, 50, 100, 1000 heterogeneous tasks.

Conclusion and Future works

Combination of two standard algorithms was taken in to account and a anew hybrid algorithm was introduced in this article named as PWOA algorithm, which is of meta-heuristic type. The PWOA algorithm has been evaluated in a range of different workflows, including Cybershake, Montage, Epigenomics, Sipht, and Inspiral. TET and TEC were the main motive which were to be reduced via this algorithm. The new algorithm stood better in comparison to standard WOA and PSO algorithms. The average TEC and TET of a single workflow (Cybershake, Epigenomics, Inspiral, Montage, Sipht) is calculated with the average total number of task results (30, 50, 100, and 1000).

The average of TEC in Cybershake, Epigenomics, Inspiral, Montage, and Sipht workflows is reduced by 22.60%, 2.46%, 6.08%, 1.23%, and 8.64% respectively as compared to standard PSO algorithm. When the new algorithm is compared to the existing standard WOA

algorithm, the average TEC in Cybershake, Epigenomics, Inspiral, Montage, and Sipht workflow is reduced by 71.31%, 56.92%, 35.24%, 35.56%, and 30.43% respectively.

Similarly, the average TET in Cybershake, Epigenomics, Inspiral, Montage, and Sipht workflows is reduced by 7.73%, 6.60%, 3.72%, 4.39%, 1.45% respectively when they were compared with PSO. When standard WOA algorithm is compared with the proposed algorithm, the average TET in Cybershake, Epigenomics, Inspiral, Montage, and Sipht workflows is reduced by 17.75%, 50.37%, 48.45%, 48.06%, and 12.47% respectively.

The proposed algorithm provides better heterogeneous results in terms of TET and TEC when compared to the standard PSO and WOA algorithms in our simulation. In future study, various parameters including total execution energy, latency, offloading, deadline, and other factors will be considered for evaluation. You may also consider other algorithms to make a new hybrid algorithm.

Declarations

Ethical Approval

There is no subject which uses human or animal data so consent is not appliable. I promise to abide rules and regulations of my society, community and country. We give our consent in favour of publisher to publish this manuscript which can include figures, tables and details within the manuscript.

Competing interests

We declare that there is no conflict of interest with anybody.

Authors' contributions

I, Sumit Bansal, wrote this manuscript for Ph.D. work and Dr. Himanshu Aggarwal reviewed and analysed the results of this manuscript.

Funding

It is non-funded research and persuaded for Ph.D. degree.

Availability of data and materials

Research is in algorithmic in nature and no data has been used.

References

1. Nair, Gnaneswar, G Hadresh, V. P. (2019) A Comparison Analysis of Fog And Cloud Computing. Int. J. Res. Anal. Rev. 6, 1386–1390x. doi: 10.31221/osf.io/y7db2.
2. Bansal, S., Aggarwal, M. & Aggarwal, H. (2019) Advancements and applications in fog computing. in Security Designs for the Cloud, IoT, and Social Networking 207–240. doi:10.1002/9781119593171.ch14.
3. Deelman, E. et al. (2015) Pegasus, a workflow management system for science automation. Futur. Gener. Comput. Syst. 46, 17–35. doi: 10.1016/j.future.2014.10.008.
4. Graves, R. et al. (2011) CyberShake: A Physics-Based Seismic Hazard Model for Southern California. Pure Appl. Geophys. 168, 367–381. doi: 10.1007/s00024-010-0161-6.

5. Jiang, Q., Lee, Y. C., Arenaz, M., Leslie, L. M. & Zomaya, A. Y. (2014) Optimizing scientific workflows in the cloud: A montage example. Proc. - 2014 IEEE/ACM 7th Int. Conf. Util. Cloud Comput. UCC 2014 517–522. doi:10.1109/UCC.2014.77.
6. Kalra, M. & Singh, S. (2019) Multi-criteria workflow scheduling on clouds under deadline and budget constraints. Concurr. Comput. Pract. Exp. 31, 1–16. doi: 10.1002/cpe.5193.
7. Diane E. Handy, Rita Castro, J. L. (2013) Epigenetic Modifications: Basic Mechanisms and Role in Cardiovascular Disease. Early Hum. Dev. 83, 1–11. doi: 10.1016/j.earlhumdev.2006.05.022.
8. Livny, J., Teonadi, H., Livny, M. & Waldor, M. K. (2008) High-throughput, kingdom-wide prediction and annotation of bacterial non-coding RNAs. PLoS One 3(9). doi: 10.1371/journal.pone.0003197.
9. Pandey, S., Wu, L., Guru, S. M. & Buyya, R. (2010) A Particle Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments. 3–10. doi:10.1109/AINA.2010.31.
10. Bansal, S., Aggarwal, H. & Aggarwal, M. (2022) A systematic review of task scheduling approaches in fog computing. Trans. Emerg. Telecommun. Technol. 33:4523. doi: 10.1002/ett.4523.
11. Ahmed, O. H. et al. (2020) Scheduling of scientific workflows in multi-fog environments using markov models and a hybrid salp swarm algorithm. IEEE Access 8, 189404–189422. doi: 10.1109/ACCESS.2020.3031472.
12. Subramoney, D. & Nyirenda, C. N. (2020) A Comparative Evaluation of Population-based Optimization Algorithms for Workflow Scheduling in Cloud-Fog Environments. in 2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020 760–767. doi:10.1109/SSCI47803.2020.9308549.
13. Liu, Z., Dai, P., Xing, H., Yu, Z. & Zhang, W. (2021) A Distributed Algorithm for Task Offloading in Vehicular Networks With Hybrid Fog/Cloud Computing. IEEE Trans. Syst. Man, Cybern. Syst. 1–14. doi:10.1109/TSMC.2021.3097005.
14. Memari, P., Mohammadi, S. S., Jolai, F. & Tavakkoli-Moghaddam, R. (2022) A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture. J. Supercomput. 78:93–122. doi: 10.1007/s11227-021-03868-4.
15. Rana, N., Abd Latiff, M. S., Abdulhamid, S. M. & Misra, S. (2021) A hybrid whale optimization algorithm with differential evolution optimization for multi-objective virtual machine scheduling in cloud computing. Eng. Optim. 1–18. doi:10.1080/0305215X.2021.1969560.
16. Arora, N. & Banyal, R. K. (2021) A Particle Grey Wolf Hybrid Algorithm for Workflow Scheduling in Cloud Computing. Wireless Personal Communications (Springer US). doi:10.1007/s11277-021-09065-z.
17. Saoud, A. & Recioui, A. (2022) Hybrid algorithm for cloud-fog system based load balancing in smart grids. Bull. Electr. Eng. Informatics 11: 477–487. doi: 10.11591/eei.v11i1.3450.
18. Thennarasu, S. R., Selvam, M. & Srihari, K. (2021) A new whale optimizer for workflow scheduling in cloud computing environment. J. Ambient Intell. Humaniz. Comput. 12:3807–3814. doi: 10.1007/s12652-020-01678-9.
19. Bisht, J. & Vampugani, V. S. (2021) Load and Cost-Aware Min-Min Workflow Scheduling Algorithm for Heterogeneous Resources in Fog, Cloud, and Edge Scenarios. Int. J. Cloud Appl. Comput. 12:1–20. doi: 10.4018/ijcac.2022010105.
20. Arora, N. & Banyal, R. K. (2021) Workflow scheduling using particle swarm optimization and gray wolf optimization algorithm in cloud computing. Concurr. Comput. Pract. Exp. 33:1–16. doi: 10.1002/cpe.6281.

21. Alsmady, A., Al-Khraishi, T., Mardini, W., Alazzam, H. & Khamayseh, Y. (2019) Workflow scheduling in cloud computing using memetic algorithm. 2019 IEEE Jordan Int. Jt. Conf. Electr. Eng. Inf. Technol. JEEIT- Proc. 302–306. doi:10.1109/JEEIT.2019.8717430.
22. Ijaz, S., Munir, E. U., Ahmad, S. G., Rafique, M. M. & Rana, O. F. (2021) Energy-makespan optimization of workflow scheduling in fog–cloud computing. Computing 103:2033–2059. doi: 10.1007/s00607-021-00930-0.
23. James Kennedy and Russell Eberhart (1995) Particle Swarm Optimization. IEEE Int. Conf. neural networks 4:1942–1948. doi: 10.1002/9780470612163.
24. Mirjalili, S. & Lewis, A. (2016) The Whale Optimization Algorithm. Adv. Eng. Softw. 95:51–67. doi: 10.1016/j.advengsoft.2016.01.008.
25. Liu, X. F. et al. (2018) An Energy Efficient Ant Colony System for Virtual Machine Placement in Cloud Computing. IEEE Trans. Evol. Comput. 22:113–128. doi: 10.1109/TEVC.2016.2623803.
26. Sahni, J. & Vidyarthi, P. (2018) A Cost-Effective Deadline-Constrained Dynamic Scheduling Algorithm for Scientific Workflows in a Cloud Environment. IEEE Trans. Cloud Comput. 6:2–18. doi: 10.1109/TCC.2015.2451649.