International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7181
***Article Received**: 22 April 2023 **Revised:** 12 June 2023 **Accepted:** 27 June 2023*

Effective Workflow Scheduling in Cloud using Constriction Factor based Inertia Weight Particle Swarm Optimization

Vinay Kumar Sriperambuduri[1] , Nagaratna M[2]
Department of CSE
[1,2]JNTUH College of Engineering
Hyderabad, India.
[1]s.vinaykumar@staff.vce.ac.in, [2]mratnaraju@jntuh.ac.in

Abstract—Cloud computing allows rapid provision of resources based on the need. This enables users to execute the independent tasks and dependent tasks called workflows on the cloud system. Workflow scheduling is a crucial problem that is NP Hard and is still a challenging problem. Particle Swarm Optimization (PSO) is one of the commonly used metaheuristic algorithms for solving task scheduling problems, but it has issues with premature convergence and lack of diversity. In recent years, chaotic maps have been employed in PSO to enhance its performance. This study proposes a Constriction factor-based inertia weight in PSO for workflow scheduling (CFPSO). The proposed algorithm utilizes a constriction factor for updating the inertia weight, which enhances the exploration ability of the algorithm thereby avoid local optima. The algorithm considers a fitness function with an aim to minimize makespan, service cost, and maximize load balance. The proposed algorithm is evaluated using a set of benchmark workflows, and the obtained results are compared with the standard PSO algorithm, Grey Wolf Optimizer (GWO) algorithm and Chaotic PSO algorithm. The extensive experimentation performed show that the proposed algorithm outperforms the other algorithms in terms of makespan, service cost, and load balance. The proposed CFPSO shows reduction of 20% of makespan, 2% of the service cost and 18% load balance rate compared to the conventional algorithms on Montage workflow with 1000 tasks. The use of constriction factor enhances the performance of the algorithm and makes it suitable for solving complex problems with multiple objectives. The proposed algorithm can be used in real-world applications to optimize workflow scheduling in cloud computing environments.

Keywords- Cloud computing; Particle swarm optimization; Grey wolf optimization; Workflow scheduling.

I. INTRODUCTION

Cloud computing has revolutionized the way computing resources are utilized and managed. Workflow scheduling is a significant issue in cloud computing that has an impact on the performance and cost of the applications. The goal of workflow scheduling is to assign tasks to resources in a way that minimizes the makespan (the time to complete all tasks), reduces the service cost, and maximizes the utilization of resources. However, due to multiple objectives and constraints this is a challenging optimization problem. Workflow scheduling architecture is described in Figure 1.

Particle Swarm Optimization (PSO) proposed by Kennedy and Eberhart [1] is a widely used technique for solving optimization problems, including workflow scheduling. PSO is a swarm-based algorithm that mimics the social behaviour of birds or other animals. It has been utilized effectively in many real-world optimization problems, including workflow scheduling. However, the standard PSO algorithm may suffer from premature convergence and lack of diversity, which can result in suboptimal solutions.

In recent years, chaotic maps have been used to enhance the performance of optimization algorithms. Chaotic maps can generate pseudo-random sequences that have high sensitivity to initial conditions and exhibit complex behaviour. This property of chaotic maps enhances the exploration ability of optimization algorithms and avoid getting stuck in local optima.

As task scheduling in cloud computing is an NP hard problem, to address the problem many optimization algorithms are developed [2-3]. The classic metaheuristics-based optimisation techniques typically take more time to compute. [4]. Additionally, the best solution can only be found by examining a bigger search region, thus the workflow in this situation needs to be well-managed [5-6].

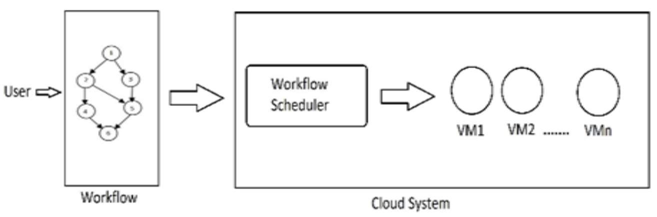


Figure 1. Workflow Scheduling architecture

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7181
***Article Received**: 22 April 2023 **Revised:** 12 June 2023 **Accepted:** 27 June 2023*

In this work, we propose a new constriction factor based PSO algorithm to perform workflow scheduling in cloud computing. The algorithm employs a constriction factor to update the inertia weight of PSO, which enhances the exploration ability of the algorithm. The fitness function considers multiple objectives, including makespan, service cost, and load balance, to find an optimal schedule. The work analysed the effectiveness of the algorithm on scientific workflows montage and sipht. The developed algorithm outperforms the other algorithms in terms of makespan, service cost or cost and load balance.

II. RELATED WORKS

The Particle swarm optimization (PSO) is a widely used metaheuristic algorithm for solving complex optimization problems. However, standard PSO tends to converge to local optima, which can limit its performance. To overcome this limitation, researchers have proposed various modifications to the algorithm, one of which is the incorporation of chaos theory. Chaotic inertia weight PSO (CPSO) algorithms employ chaotic sequences to introduce randomness into the search process, which can help the algorithm avoid local optima and improve its performance. In this work we propose to introduce Constriction factor-based inertia weight to avoid local optima.

The study by [7] proposed an adaptive PSO algorithm for scheduling workflows in the cloud environment. The technique uses a method depending on the type of VM and the amount of its consumption. A given VM type's number is increased by 10% or dropped by 10% depending on its consumption. The results showed that the proposed algorithm outperformed traditional PSO in terms of makespan and cost.

Similarly, [8] proposed a chaotic PSO (CPSO) for workflow scheduling to minimize cost and avoid premature convergence. The chaotic sequence applied is irregular and unpredictable that has helped improve global search. They also inducted a penalty factor for deadline is skipped and for the VMs idle time. The results showed that the proposed algorithm outperformed traditional PSO in terms of cost. However, load balance is ignored.

In [9] proposed a novel adaptive inertia weight based PSO that utilizes feedback on the swarm's success rate to determine the particles' state in the search space. The algorithm is evaluated on static test problems and compared against other inertia weight techniques involving fuzzy rules, time-varying and random. They did not consider tasks or workflow for evaluation. Another study by Prasanna et al. (2020) [10] proposed a PSO algorithm with a greedy algorithm and clustering technique to optimize the makespan in task scheduling. The results are compared with Greedy PSO and showed that the cost is minimized. The proposed algorithm did not take dependent tasks in to consideration. In optimization problems, Wang et al. (2021) [11] proposed a hybrid PSO algorithm that uses chaotic map and adaptive strategy to optimize melt spinning progress application. The proposed algorithm demonstrated better performance compared to other algorithms such as ant colony optimization and genetic algorithms. The work did not focus on task scheduling. In [12] authors proposed a hierarchical scheduling strategy to minimize the cost. They have considered the service contract signed between the vendor and the user. They tested the algorithm on SwinDeW-C architecture. In [13], authors developed a variable neighborhood search technique as part of PSO algorithm that minimizes the total cost of a workflow.

Li et. al. in [14] viewed the structure of the workflow as a control structure and applied reduction technique. The results showed the reduction of the execution cost in large workflows. However, the total execution time and load were not considered.

In [15] authors proposed the assignment of tasks to VMs based on the Poset with emphasis on critical and non-critical paths to optimize cost and response time. Load balance is an important aspect to be considered.

Many heuristic algorithms [16] were proposed, such as PSO [17,18,19] for workflow scheduling, Ant colony optimization [20,21] and Genetic Algorithm [22,23] to solve task scheduling problems. In [24], Mirjalili et. al. proposed a GWO algorithm that can be applied to optimization problems including workflow scheduling.

To further improve exploration capability and to minimize makespan and cost and maximize load balance on the resources the proposed approach utilizes constriction factor-based inertia weight.

III. PROPOSED METHODOLOGY

In this manuscript, the effectiveness of the proposed CFPSO algorithm is investigated on the Montage and Sipht workflows. One of the astronomy applications produced by the NASA/IPAC Centre is the Montage Workflow. Sipht is a bioinformatics-based application developed by Harvard

A. Problem Definition

A workflow refers to a sequence of tasks that need to be executed on a set of machines to achieve a specific goal. Workflows typically involve many tasks that are interdependent and need to be executed in a definite order to achieve the desired outcome. A workflow W= (G,E) represented as DAG as shown in Figure 2 with G as a set of nodes and E as edges that show the dependencies between the nodes or tasks.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7181
***Article Received**: 22 April 2023 **Revised:** 12 June 2023 **Accepted:** 27 June 2023*

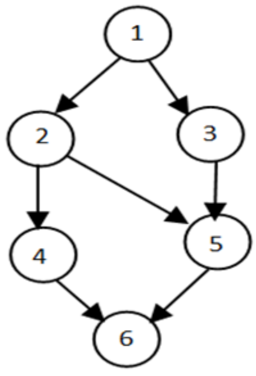


Figure 2. Representation of sample workflow

The workflow tasks are computationally intensive or data intensive in nature. A cloud vendor maintains a data center to support the clients in terms of compute, network and storage services. Data center, houses several host machines that provide Virtual machines (VMs) as resources to the clients. The workflow is submitted to the scheduler as input to generate a mapping for Tasks to VMs ensuring the optimization of defined parameters. In the proposed algorithm the parameters such as Makespan (MS) , cost (C) and Load balance (LB) are optimized.

Makespan is measured from the begin time of the initial task in the workflow to the finish time of the final task. It is presented in Eq. (1).

$$Makespan = Maximum\{FinishTime(t1, t2 \dots tn)\} \quad (1)$$

Service Cost or Cost of executing a workflow on resources is given in Eq. (3). It is computed based on the parameters such as execution time of the tasks, data-transfer time, price of a VM and data-transfer cost. The execution time of a task is presented in Eq. (2).

$$E_{ti} = \frac{Length\ of\ the\ task\ in\ MIPS}{VM\ capacity\ in\ MIPS} \quad (2)$$

$$Cost = \sum_{i=1}^{n}\left((E_{ti} \times C_{vmj}) + (\frac{dt_i}{BW} \times C_{BW})\right) \quad (3)$$

Where

E_{ti} refers to the execution time of the task i

C_{vmj} refers to the cost of a VM j and j ranges between $1 \leq j \leq Max. No. of\ VMs$

dt_i refers to the size of the data given as input to task i.

C_{BW} refers to the bandwidth cost.

Load balance refers to utilizing all the resources to equal extent ensuring resource overloading. We have applied the Load Balance Rate (LBR) mentioned in Eq. (4) as the measure to determine the load balance across the available resources. LBR value =1 means the resources are well balanced or equally balanced, this being the ideal case. The LBR value more than 1 indicates an imbalance in utilization of the resources.

$$Load\ balance\ rate = \left(\frac{MRU}{ARU}\right) \quad (4)$$

MRU = Max. resource usage across all the resources used in executing the workflow

ARU = Average resource usage across all the resources used in executing the workflow.

The Fitness calculation is performed taking in to account three parameters cost, makespan and load balance and is represented in Eq. (5).

$$Fitness = \alpha \times \left(\frac{1}{Makespan}\right) + \beta \times \left(\frac{1}{Cost}\right) - \gamma \times \left(\frac{1}{Load\ balance\ rate}\right) \quad (5)$$

α, β, γ represent the weightage given to each of the parameters. The current work considers α=0.4, β=0.3 and γ=0.3.

B. Particle Swarm Optimization Algorithm

Particle Swarm Optimization (PSO) developed by Kennedy and Eberhart [1] is a swarm-based algorithm that is inspired by flocks of birds, where each individual in the group follows the movement of its neighbors to find the optimal path towards a goal. Below are the steps of the algorithm.

```
PSO Algorithm
1. Initialization of algorithmic parameters and the population
   of particles with position
2. Calculate fitness of each particle
3. Evaluate fitness value
4. Update particle best and global best positions
5. Update particle velocity and position
6. Repeat steps 2 to step 5 until maximum iterations is
   completed
```

PSO uses a set of particles or population to search through the search space for the best possible solution. The location of each particle in the search space corresponds to a potential solution here it represents the task to VM mappings, and its velocity denotes the speed and direction of movement. Through the exchange of knowledge about their current optimum placements, the particles in the population communicate with one another. This interaction is made possible through a process of collective learning in which each particle modifies its velocity according to its individual experience called personal best and the collective wisdom of the swarm called the global best.

Particles move around the search space while the optimization process is underway by varying their velocities in accordance with acceleration constants, random variables, and the distance between their present places and the optimal placements discovered. This makes it possible for particles to find their way towards promising areas of the search space and converge on the best ones.

Until a termination requirement, like reaching a maximum number of iterations, is fulfilled, the algorithm iteratively updates the positions and velocities of the particles. The best position found by the swarm represents the optimal solution to the problem.

C. Grey Wolf Optimizer Algorithm (GWO)

Mirjalili et.al. [19] observing the behaviour of grey wolves proposed an algorithm named GWO. Wolf pack consists of α-wolf, β-wolf, δ-wolf and ω-wolves. α-wolf, β-wolf, δ-wolf are the top three wolves in order of precedence and all the remaining wolves called ω-wolves follow these three wolves. Wolf pack follows a series of steps as represented in the below mentioned algorithm to attack a prey.

GWO Algorithm

```
1.  Initialize a population of grey wolves randomly within the search space.
2.  Define the fitness value for each wolf based on the objective function.
3.  Identify the three alpha, beta, and delta wolves with the highest fitness values in the population.
4.  **While** maximum number of iterations:
5.  Update the positions of all the wolves using the following equations:
6.  **For** each wolf:
7.  Update the position of the wolf by adjusting it towards the alpha, beta, and delta wolves using the hunting equation.
8.  Limit the new position within the bounds of the search space.
9.  Update the fitness values for the new positions of the wolves.
10. **End For**;
11. Identify the new alpha, beta, and delta wolves based on their fitness values.
12. **End While**
13. Return the best wolf (alpha) and its corresponding fitness value as the output.
```

To update the the positions of the wolves, the following hunting equations represented by Eq. (6), (7) and (8) are used for each wolf. The new position is computed using the Eq. (9).

$$\overrightarrow{D_\propto} = \left|\overrightarrow{C_1} * \overrightarrow{X_\alpha} - \vec{X}\right|$$

$$\overrightarrow{X_1} = \overrightarrow{X_\propto} - \overrightarrow{A_1} . (\overrightarrow{D_\propto}) \qquad (6)$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2} * \overrightarrow{X_\beta} - \vec{X}\right|$$

$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} . (\overrightarrow{D_\beta}) \qquad (7)$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3} * \overrightarrow{X_\delta} - \vec{X}\right|$$

$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} . (\overrightarrow{D_\delta}) \qquad (8)$$

$$\overrightarrow{X_{new}} = \frac{\overrightarrow{X_1}+\overrightarrow{X_2}+\overrightarrow{X_3}}{3} \qquad (9)$$

where X represents the position of the wolf, X_newis the new position after the updation, X_∝,X_β,X_δ are the positions of the α, β, γ wolves, respectively, C_1 and C_2 are coefficients that alter the exploration and exploitation ability of the algorithm.

The equations for scaling factor A ⃗ and C ⃗ are given by Eq. (10) and eq. (11).

$$\vec{A} = (2 \times \vec{a} \times \overrightarrow{r_1}) - \vec{a} \qquad (10)$$

$$\vec{C} = (2 \times \overrightarrow{r_2}) \qquad (11)$$

$\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vectors =[0,1].

D. Proposed Constriction Factor based Particle Swarm Optimization Algorithm (CFPSO)

The proposed algorithm initializes the population of particles, where each particle represents a potential solution referring to the allocation of tasks to VMs. These particles move through the search space by updating the position and velocity of each particle based on its own experience called as personal best and the experience of the swarm called as global best. For each ith particle the velocity and position are updated using the equations represented by Eq. (12) and Eq. (13) respectively as shown below.

$$V_i^{t+1} = \omega . V_i^t + c_1 . r_1 . \left(X_{pbest,i}^t - X_i^t\right) + c_2 . r_2 . \left(X_{gbest,i}^t - X_i^t\right) \qquad (12)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \qquad (13)$$

Where,

V_i^t = Velocity of the particle i at iteration t

V_i^{t+1} = Velocity of particle i at iteration t+1

ω = Inertia weight

c_1 & c_2 = Acceleration coefficients

r_1 & r_2 = Random numbers in range [0,1]

X_i^t = Current position of particle i at iteration t

$X_{pbest,i}^t$ =Personal best position of particle i at iteration t

$X_{gbest,i}^t$ = Global best position of the particle in a population

X_i^{t+1} =Position of the particle i at iteration t+1.

Each particle based on its new position is evaluated against the fitness function as shown in Eq. (5) to optimize makespan, cost and load balance.

Proposed Algorithm CFPSO

```
Input:
  - objective function f(x)
  - population of size M
  - maximum number of iterations MAXiter
  - initial swarm position x_i and velocity v_i for
    i = 1, 2,..,M
  -maximum and minimum velocity limits v_max
    and v_min
  -maximum and minimum position limits x_max
    and x_min
  - constriction factor - initial inertia weight ω
  - parameters c1 and c2 (acceleration constants)
  - parameters ω_max and ω_min (maximum
    and minimum inertia weights)
Output:
  - the best position found x_best
  - the best fitness value found f_best
1. Initialize the particle positions and velocities for each particle in the swarm.
2. Initialize personal best positions and objective (fitness) values for each particle.
3. Initialize the global best position.
4. Initialize the current iteration number.
5. Initialize the inertia weight ω.
6. Perform iterations until the maximum number of iterations is reached.
7.      For each particle in the swarm:
    a) Generate random values r1 and r2.
    b) Update the velocity of the particle using the PSO equation using Eq. (12), considering its previous velocity, personal best position, and global best position.
    c) Clip the velocity within the maximum and minimum velocity limits.
    d) Update the position of the particle using the new velocity using Eq. (13).
    e) Clip the position within the maximum and minimum position limits.
    f) Evaluate the fitness value of the new position using Eq. (5).
    g) If the fitness value is better than the personal best, update the personal best position and fitness value.
    h) If the fitness value is better than the global best, update the global best position and fitness value.
8. Update the constriction factor φ using Eq. (14).
9. Update the inertia weight ω using Eq. (15).
10. Increment the iteration number.
11. Output the best position found and the best fitness value found
```

The proposed algorithm CFPSO employs the constriction factor (φ) as indicated in Eq. (14) to update the inertia weight (ω), which controls the exploration and exploitation of search space in a balanced manner in the algorithm. A Constriction factor was introduced by Clerc [25] in 1999 in his study on convergence and stability of PSO. It is a scaling parameter that restricts the velocity of particles during the PSO optimization process. The idea behind the constriction factor is to ensure that particles converge to the global optimum in a stable and efficient manner, while also preventing them from overshooting the optimal solution.

$$\varphi = \frac{2}{\left|2-c_1-c_2-\sqrt{{c_1}^2+{c_2}^2+2c_1c_2-2c_1-2c_2+4}\right|} \quad (14)$$

where c_1 and c_2 are the acceleration coefficients used in the velocity update equation and ensures the values is in the range [0,1].

The inertia weight computation is done using the below Eq. (15). This computation utilizes constriction factor (φ).

$$\omega = \frac{\omega_{max}-\omega_{min}}{\varphi} + (\varphi\ (\omega_{n-1} - \omega_{max}) + \omega_{max}) \quad (15)$$

Where,

ω_{max} = Maximum Inertia Weight Wight

ω_{min} = Minimum Inertia Weight

ω_{n-1}= Inertia Weight generated in the previous iteration.

E. Case Study

Consider a workflow with 6 tasks shown in Figure 3 as shown below.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7181
***Article Received**: 22 April 2023 **Revised:** 12 June 2023 **Accepted:** 27 June 2023*

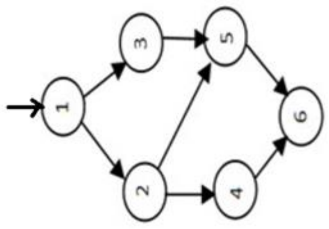


Figure. 3 Workflow for casestudy

a) Particle Swarm Optimization (PSO): The Gantt chart for the workflow execution on PSO algorithm is shown in Figure 4. The figure clearly indicates the makespan for PSO is 130.55. The load balance rate is 1.20 and cost of the workflow as per the Eq. (3) is 357.43.

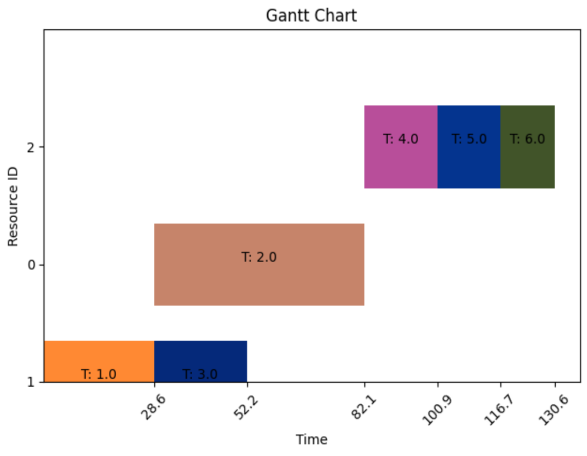


Figure 4. Gantt chart for workflow execution with PSO

The Load Balance Rate as per Eq. (4) is given below.

Load Balance Rate = MRU/ARU=51.7/42.96=1.20

b) Grey Wolf Optimizer: The Gantt chart for the workflow execution on PSO algorithm is shown in Figure 5. The figure clearly indicates the makespan for GWO as 125.1. The cost and load balance rate are 362.91 and 1.07 respectively.

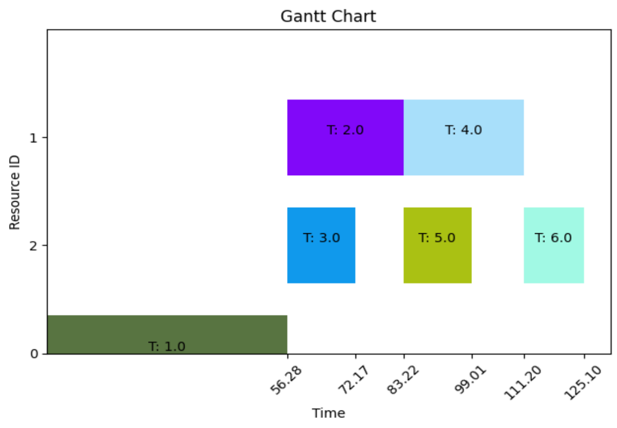


Figure 5. Gantt chart for workflow execution with GWO

c) Chaotic PSO(CPSO): The Gantt chart for the workflow execution on CPSO algorithm is shown in Figure 6. The figure indicates the makespan using CPSO is 119.65. The cost of the workflow is 357.43 and load balance rate is 1.05.

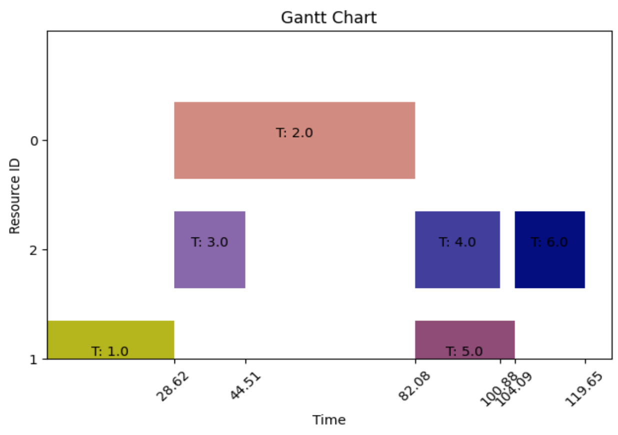


Figure 6. Gantt chart for workflow execution with CPSO

d) Constriction Factor based PSO (CFPSO): The Gantt chart for the workflow execution on CFPSO algorithm is shown in Figure 7. The figure clearly indicates the makespan for PSO is 113.01. The cost of the workflow is 347.47 and Load balance rate is 1.04.

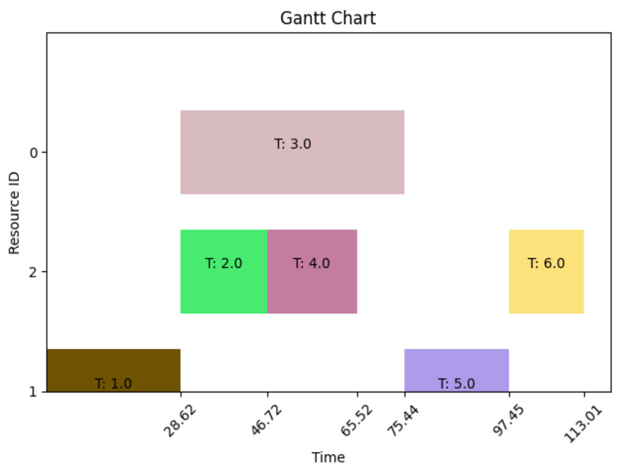


Figure 7. Gantt chart for workflow execution with CFPSO

IV. SIMULATION RESULTS

The proposed algorithm is executed on a system with Core i5 processor, 16.00 GB RAM, Windows 8, 64-bit operating system. The results are compared against regular PSO algorithm and chaotic PSO algorithm for two scientific workflows Montage and Sipht. These workflows are available in different sizes, for example, Montage workflows contain 25, 50, 100, & 1000 tasks and Sipht workflows contain 30,60, 100 & 1000 tasks. The simulation and evaluation of performance of the proposed algorithms is carried out on WorkflowSim-1.0, which is an extension to CloudSim.

***Article Received**: 22 April 2023 **Revised:** 12 June 2023 **Accepted:** 27 June 2023*

The set of values used for the parameters in the algorithms to perform an experimentation to optimize makespan, service cost and load balance is indicated in Table 1.

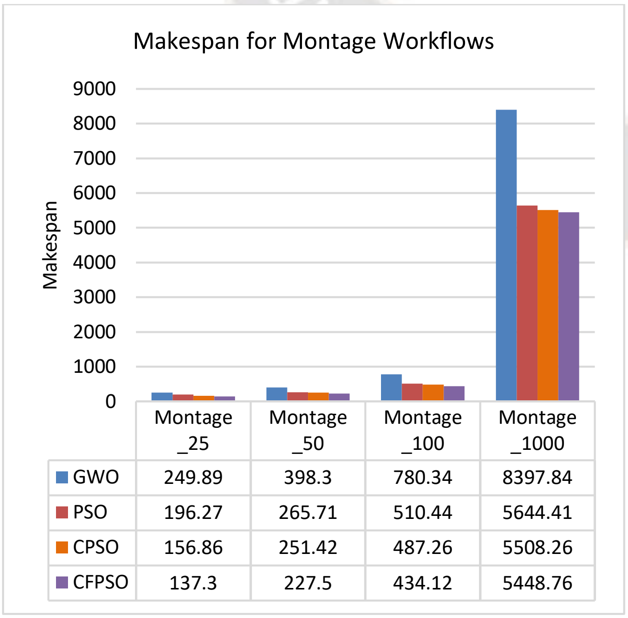
A. Makespan Analysis

The makespan comparison of CFPSO is done with GWO, PSO and CPSO as shown in Figure 8 for Montage workflows and Figure 9 for Sipht workflows. The makespan improvement over GWO is 82%, over PSO is 42.95% and over CPSO algorithm is 14.25% for Montage workflow with 25 tasks. For Montage 50, it is 75.08%, 16.80% and 10.51%, for Montage 100, the improvement is 79.75%, 17.58% and 12.24%. Whereas for Montage 1000, the improvement is 54.12%, 3.59% and 1.09% over GWO, PSO and CPSO respectively.

TABLE I. PARAMETERS SET FOR ALGORITHMS

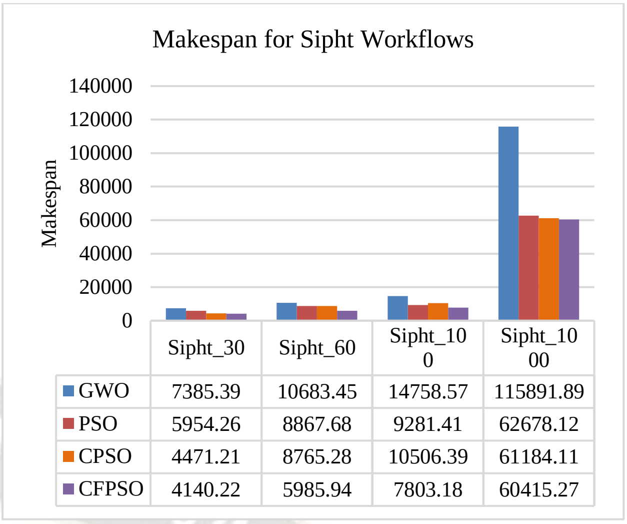
PARAMETERS	VALUES
NUMBER OF TASKS	25-1000
NUMBER OF PARTICLES	100
NUMBER OF ITERATIONS	500
R1, R2	RANDOM [0,1]
C1, C2	0.9
Ω	0.1 (PSO), CHAOTIC INERTIA (CPSO), CONSTRICTION FACTOR INERTIA (CFPSO)
NUMBER OF VMS	5 (HETEROGENOUS)
BANDWIDTH	100

For Sipht workflow, the improvement is over 78.38%, 43.82%, 7.99% for 30 tasks, 78.48%, 48.14%, 46.43% for 60 tasks, 89.14%, 18.94%, 34.64% for 100 tasks and 91.83%, 3.75%, 1.27% for 1000 tasks over GWO, PSO and CPSO.



	Montage _25	Montage _50	Montage _100	Montage _1000
🟦 GWO	249.89	398.3	780.34	8397.84
🟥 PSO	196.27	265.71	510.44	5644.41
🟧 CPSO	156.86	251.42	487.26	5508.26
🟪 CFPSO	137.3	227.5	434.12	5448.76

Figure 8. Makespan for Montage Workflows

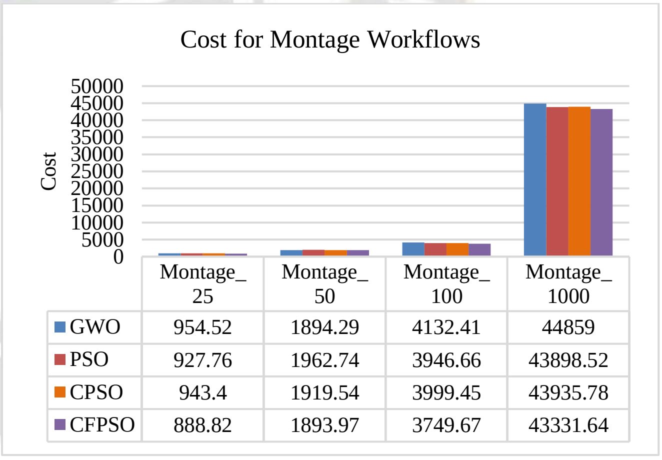


	Sipht_30	Sipht_60	Sipht_100	Sipht_1000
🟦 GWO	7385.39	10683.45	14758.57	115891.89
🟥 PSO	5954.26	8867.68	9281.41	62678.12
🟧 CPSO	4471.21	8765.28	10506.39	61184.11
🟪 CFPSO	4140.22	5985.94	7803.18	60415.27

Figure 9. Makespan for Sipht workflows

B. Service Cost Analysis

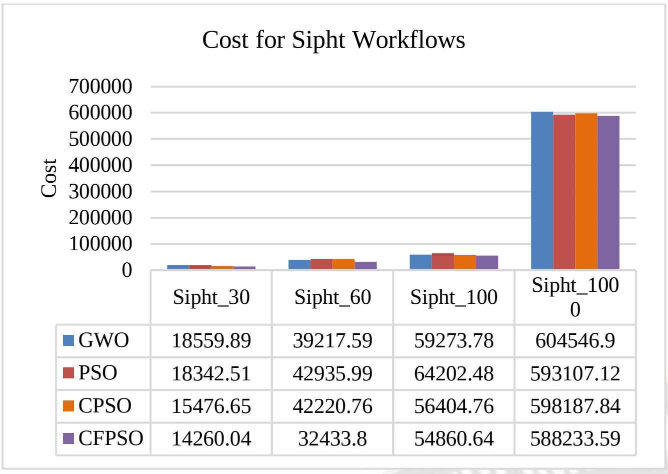
In case of Montage workflow, the service cost through CFPSO against GWO, PSO, CPSO is showing 7.39%, 4.38%, 6.14% increase for 25 tasks, 0.02%, 3.63%, 1.35% increase for 50 tasks, 10.21%, 5.25%, 6.66% increase for 100 tasks and 3.52%, 1.31%, 1.39% increase for 1000 tasks as shown in Figure 10. For Sipht workflow, it is 30.15%, 28.63%, 8.53% increase for 30 tasks, 20.92%, 32.38%, 30.18% increase for 60 tasks, 8.04%, 17.03%, 2.81% increase for 100 tasks and 2.77%, 0.83%, 1.69% increase for 1000 tasks as shown in Figure 11.



	Montage_ 25	Montage_ 50	Montage_ 100	Montage_ 1000
🟦 GWO	954.52	1894.29	4132.41	44859
🟥 PSO	927.76	1962.74	3946.66	43898.52
🟧 CPSO	943.4	1919.54	3999.45	43935.78
🟪 CFPSO	888.82	1893.97	3749.67	43331.64

Figure 10. Cost for Montage workflows

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7181
***Article Received**: 22 April 2023 **Revised:** 12 June 2023 **Accepted:** 27 June 2023*



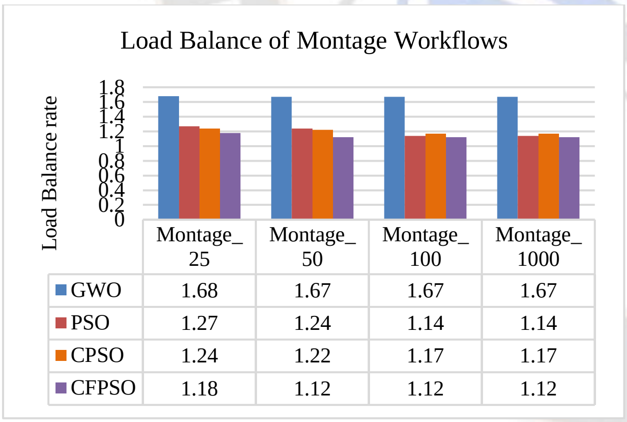
	Sipht_30	Sipht_60	Sipht_100	Sipht_1000
🟦 GWO	18559.89	39217.59	59273.78	604546.9
🟥 PSO	18342.51	42935.99	64202.48	593107.12
🟧 CPSO	15476.65	42220.76	56404.76	598187.84
🟪 CFPSO	14260.04	32433.8	54860.64	588233.59

Figure 11. Cost for Sipht workflows

The minimum load rate value indicates a good balance of resources, ideally the load rate should be equal to 1. The average load rate by proposed algorithm shows 6%, 2% and 3% increase over the existing algorithms for Montage, Epigenomics and Sipht workflows. Figure 8 represent the improvement in load rate which is an indicator of load balance on resources.

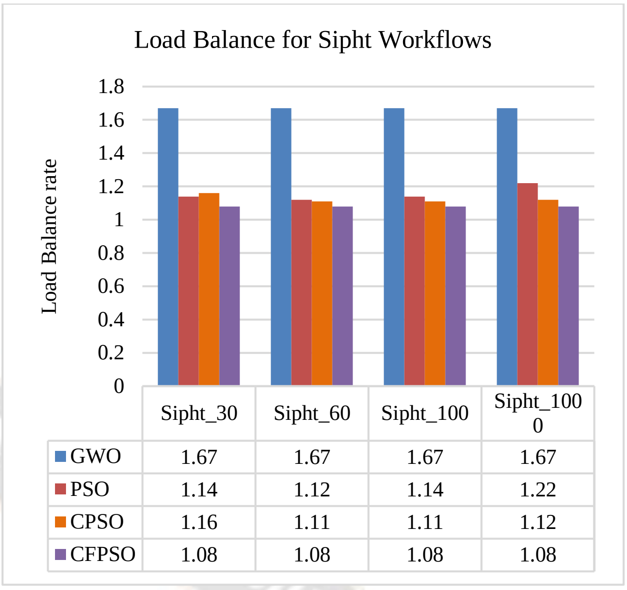
C. *Load Balance Analysis*

The average load balance is improved by 47.42%, 5.48%, 5.74% over GWO, PSO and CPSO for montage workflows and 54.63%, 6.94%, 4.17% for Sipht workflows respectively as shown in Figure 12, Figure 13 respectively.



	Montage_25	Montage_50	Montage_100	Montage_1000
🟦 GWO	1.68	1.67	1.67	1.67
🟥 PSO	1.27	1.24	1.14	1.14
🟧 CPSO	1.24	1.22	1.17	1.17
🟪 CFPSO	1.18	1.12	1.12	1.12

Figure 12. Load balance for Montage workflows



	Sipht_30	Sipht_60	Sipht_100	Sipht_1000
🟦 GWO	1.67	1.67	1.67	1.67
🟥 PSO	1.14	1.12	1.14	1.22
🟧 CPSO	1.16	1.11	1.11	1.12
🟪 CFPSO	1.08	1.08	1.08	1.08

Figure 13. Load balance for Sipht workflows

V. CONCLUSION AND FUTURE WORK

In conclusion, the use of constriction factor-based inertia weight in PSO for optimizing makespan, service cost, and load balance in workflow scheduling has shown promising results. The use of constriction factor to generate the inertia weight enhanced the exploration and exploitation of the search space, leading to better convergence and higher-quality solutions. The proposed methods have been tested on Montage and Sipht benchmark scientific datasets and have been shown to outperform traditional optimization methods in terms of the quality of the solutions with minimal makespan, service cost, and optimal load balance. The makespan on montage workflow with 1000 tasks show that the developed CFPSO algorithm is superior compared to GWO, PSO and CPSO algorithms by 54.12%, 3.59% and 1.09%, the cost is minimized by 3.52%, 1.31% and 1.39% and the load balance is improved by 49.11%, 1.79% and 4.46% respectively. Similarly, the performance on Sipht workflows also shows improvement in terms of makespan, cost and load balance.

Future research can focus on extending the proposed methods to address other challenges in workflow scheduling, such as energy optimization and reliability. Additionally, the combination of constriction based PSO with other optimization algorithms, like grey wolf optimization, genetic algorithms and ant colony optimization, can be explored to further improve the performance of the optimization method. Overall, the use of constriction factor based PSO optimization technique in workflow scheduling shows great potential for solving complex problems in various domains.

CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
DOI: https://doi.org/10.17762/ijritcc.v11i8s.7181
***Article Received**: 22 April 2023 **Revised:** 12 June 2023 **Accepted:** 27 June 2023*

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization", Encyclopedia of Machine Learning, pp. 760–766, 2010.

[2] M. Karpagam, K. Geetha, and C. Rajan, "A modified shuffled frog leaping algorithm for scientific workflow scheduling using clustering techniques", Soft Computing, Vol. 24, No. 1, pp. 637-646, 2020.

[3] A. Mohammadzadeh, M. Masdari, F.S. Gharehchopogh, and A. Jafarian, "A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling", Cluster Computing, Vol. 24, No. 2, pp. 1479-1503, 2021.

[4] A. Kaur, P. Singh, R.S. Batth, and C. P. Lim, "Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud", Software: Practice and Experience, Vol. 52, No. 3, pp. 689-709, 2022.

[5] N. Anwar and H. Deng, "A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment", Applied sciences, Vol. 8, No. 4, p. 538, 2018.

[6] I. Gupta, M.S. Kumar, and P.K. Jana, "Efficient workflow scheduling algorithm for cloud computing system: a dynamic priority-based approach", Arabian Journal for Science and Engineering, Vol. 43, No. 12, pp. 7945-7960, 2018.

[7] Xue jun Li, Jia Xu,and Yun Yang, P. Guo and Z. Xue, "An Adaptive PSO Based Real-Time Workflow Scheduling Algorithm in Cloud Systems", 17th IEEE International Conference on Communication Technology, 2017, pp. 1932-1936.

[8] Xuejun Li, Jia Xu,and Yun Yang, "A Chaotic Particle Swarm Optimization-Based Heuristic for Market-Oriented Task-Level Scheduling in Cloud Workflow Systems", Computational Intelligence and Neuroscience, 2015.

[9] Emma Smith, Deep Learning for Gesture Recognition and Human-Computer Interaction , Machine Learning Applications Conference Proceedings, Vol 3 2023.

[10] Ahmad Nickabadi, Mohammad Mehdi Ebadzadeh, Reza Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight", Applied Soft Computing, Volume 11, Issue 4, 2011, Pages 3658-3670.

[11] Rui Wang, Kuangrong Hao, Lei Chen, Tong Wang, Chunli Jiang, "A novel hybrid particle swarm optimization using adaptive strategy", Information Sciences, Volume 579, 2021, Pages 231-250.

[12] Y. Home Prasanna Raju and Nagaraju Devarakonda, "Greedy-Based PSO with Clustering Technique for Cloud Task Scheduling", Proceedings of International Conference on Computational Intelligence and Data Engineering, pp 133–141, Dec 2020.

[13] Wu, Z., Liu, X., Ni, Z. Dong Y., Yun Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems", Journal of Super computing, 63, 256–293, 2013.

[14] Nuttapong Netjinda, Booncharoen Sirinaovakul, and Tiranee Achalakul, "Cost optimal scheduling in IaaS for dependent workload with particle swarm optimization", J. Supercomput. 68, 1579–1603, 2014.

[15] Jasim, R. M. . (2023). Hybride Particle Swarm Optimization to Solve Fuzzy Multi-Objective Master Production Scheduling Problems with Application. International Journal of Intelligent Systems and Applications in Engineering, 11(1s), 201–208. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2493

[16] H. Li, H. Liu and J. Li, "Workflow scheduling algorithm based on control structure reduction in cloud environment," 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, USA, pp. 2587-2592, 2014.

[17] M. J. Nadjafi-Arani, S. Doostali and M. Younis, "Workflow Scheduling with Guaranteed Responsiveness and Minimal Cost," in IEEE Transactions on Services Computing, 2022.

[18] Sriperambuduri Vinay Kumar, M Nagaratna, Lakshmi Harika Marrivada, "Task scheduling in cloud computing using PSO algorithm", Smart Intelligent Computing and Applications, Volume 1: Proceedings of Fifth International Conference on Smart Computing and Informatics (SCI 2021), Springer Nature, 2021.

[19] C.-H. Liu, W.-H. Huang, and P.-C. Chang, "A two-stage AIS approach for grid scheduling problems", International Journal of Production Research, vol. 50, no. 10, pp. 2665–2680, 2012.

[20] Sriperambuduri Vinay Kumar, M. Nagaratna, "Comparison of Algorithms for Workflow Applications in Cloud Computing", International Journal of Engineering and Advanced Technology (IJEAT), Volume-11 Issue-1, October 2021.

[21] W.-N. Chen and J. Zhang, "A set-based discrete PSO for cloud workflow scheduling with user-defined Qos constraints," in Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC '12), pp. 773–778, 2012.

[22] X. Li, S. Zhou, J. Wang, X. Liu, Yiwen Zhang, Cheng Z., and Yun Yang, "Time-sharing virtual machine based efficient task-level scheduling in scientific cloud workflows", Future Information Technology, pp. 121–126, Springer, 2014.

[23] P. Hirsch, A. Palfi, and M. Gronalt, "Solving a time constrained two-crane routing problem for material handling with an ant colony optimisation approach: an application in the roof-tile industry", International Journal of Production Research, vol. 50, no. 20, pp. 6005–6021, 2012.

[24] Andrew Hernandez, Stephen Wright, Yosef Ben-David, Rodrigo Costa, David Botha. Intelligent Decision Making: Applications of Machine Learning in Decision Science. Kuwait Journal of Machine Learning, 2(3). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/197

[25] Sardaraz M, Tahir M., "A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing", International Journal of Distributed Sensor Networks, 2020.

[26] Henrique Yoshikazu Shishido, Júlio Cezar Estrella, Claudio Fabiano Motta Toledo, Marcio Silva Arantes, "Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds", Computers & Electrical Engineering, Volume 69, Pages 378-394, 2018.

[27] Seyedali Mirjalili, Seyed Mohammad Mirjalili, Andrew Lewis, "Grey Wolf Optimizer", Advances in Engineering Software, Volume 69, Pages 46-61, 2014.

[28] Clerc M., "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization", Proceedings of the 1999

Congress on Evolutionary Computation, Vol. 3, pp. 1951- 1957, 1999.