*International Journal on Recent and Innovation Trends in Computing and Communication*
*ISSN: 2321-8169 Volume: 11 Issue: 9s*
*DOI: https://doi.org/10.17762/ijritcc.v11i9s.7744*
***Article Received**: 19 May 2023 **Revised:** 20 July 2023 **Accepted:** 07 August 2023*

# A Hybrid Grey Wolf Optimization and Constriction Factor based PSO Algorithm for Workflow Scheduling in Cloud

**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**—Due to its flexibility, scalability, and cost-effectiveness of cloud computing, it has emerged as a popular platform for hosting various applications. However, optimizing workflow scheduling in the cloud is still a challenging problem because of the dynamic nature of cloud resources and the diversity of user requirements. In this context, Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO) algorithms have been proposed as effective techniques for improving workflow scheduling in cloud environments. The primary objective of this work is to propose a workflow scheduling algorithm that optimizes the makespan, service cost, and load balance in the cloud. The proposed HGWOCPSO hybrid algorithm employs GWO and Constriction factor based PSO (CPSO) for the workflow optimization. The algorithm is simulated on Workflowsim, where a set of scientific workflows with varying task sizes and inter-task communication requirements are executed on a cloud platform. The simulation results show that the proposed algorithm outperforms existing algorithms in terms of makespan, service cost, and load balance. The employed GWO algorithm mitigates the problem of local optima that is inherent in PSO algorithm.

**Keywords**- Workflow, Scheduling; PSO; GWO; Cloudsim; Workflowsim; Cloud Scheduling.

## I. INTRODUCTION

Cloud computing has become an increasingly popular platform for hosting various applications due to its flexibility, scalability, and cost-effectiveness. However, optimizing workflow scheduling in the cloud is still a challenging problem due to the scale of the resources, dynamic nature of the resources and the diversity of user requirements. Workflow scheduling involves allocating resources and scheduling tasks in a way that minimizes makespan (i.e., the time it takes to complete all tasks) and reduces costs while efficiently utilizing available resources.

A workflow can be represented as a directed acyclic graph (DAG), where each node represents a task subset of Tasks (T) and edges (E) represent dependencies between tasks. The DAG is denoted as G(T, E), where T is the set of tasks and E is the set of edges. The entry task, which has no parent, and the exit task, which has no children, are two special tasks. The paired set of tasks in E indicates that the second task must be executed after the first task. For instance, if the pair {T1, T2} exists in E, then T2 is dependent on T1 and must be executed after T1.

Figure 1 shows an example of a simple workflow DAG, consisting of six tasks: T1, T 2, T3, T4, T5, T6 with T1 and T6 being the start and end tasks, respectively. The complete set of dependencies in the example DAG is {{T1, T2}, {T1, T3}, {T2, T4}, {T2, T4}, {T2, T5}, {T4, T6}, and {T5, T6}}.

The Pegasus project offers various scientific workflows, including Montage, Inspiral, Epigenomics, and Sipht, to facilitate tasks such as disaster modeling and prediction, gravitational waveform analysis, custom mosaic creation, and bioinformatics research. These workflows serve as benchmark models for comparing scheduling algorithms and provide behavioral insights on the real-world applications. Workflow scheduling is viewed as a problem to map the tasks to VMs, where tasks are assigned to available virtual machines to optimize specific objectives.

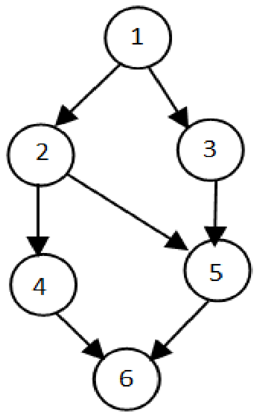

Figure 1. A Sample Workflow

To address this problem, various optimization techniques have been proposed, including meta-heuristic algorithms such as Particle Swarm Optimization (PSO), Round Robin, Heterogeneous Earliest Deadline First, Grey Wolf Optimization (GWO), and Ant Colony Optimization (ACO) in literature [1]. Each of these algorithms have shown optimizing workflow scheduling in cloud computing on specific set of objectives.

In this context, this work proposes a workflow scheduling algorithm that optimizes makespan, cost and improves load balance on resources using GWO [2] and PSO [3] algorithms in cloud computing. The algorithm is evaluated using a simulation-based approach, where a set of scientific workflows with varying task sizes and inter-task communication requirements are executed on a cloud platform.

The remaining part of this paper is organized as follows: Section II provides a review of related work on workflow scheduling in cloud computing. Section III describes the proposed workflow scheduling algorithm using PSO and GWO. Section IV presents the results of the experimentation and analysis of the proposed algorithm. Finally, Section V provides conclusion of the and the scope for future work.

## II. TYPE STYLE AND FONTS

The problem of workflow scheduling in cloud computing has received significant attention in the literature. Various techniques have been proposed to optimize workflow scheduling, including heuristics, meta-heuristics, and machine learning approaches.

PSO is a meta-heuristic optimization algorithm that is inspired by the social behavior of bird flocking. PSO has been applied to various optimization problems, including workflow scheduling in cloud computing. For example, in a study by Maria A. Rodriguez [4], PSO was used to optimize workflow scheduling in cloud computing while considering total execution cost and deadline constraints. The results show improvement in meeting the deadlines to execute workflows.

Authors in [5] proposed a novel PSO algorithm that employs a chaotic search mechanism to improve global convergence is demonstrated. They have applied the technique on various test functions to evaluate the performance.

Dynamic Non-Linear PSO [6] proposes a new inertia weight that solves the local optima problem and effectively reduce the energy consumption without considering other parameters.

GWO is another meta-heuristic optimization algorithm that is inspired by the social behavior of grey wolves. GWO has been applied to various optimization problems, including workflow scheduling in cloud computing. For example, in a study in [7], authors proposed Distributed GWO to schedule to optimize computation time and computation cost. The results showed that DGWO outperformed other meta-heuristic algorithms in terms of makespan and cost.

In [8] they proposed GWO with modification to fitness that considers makespan and cost. The algorithm is not tested on workflows with dependent tasks.

Junlong Zhou et. al. [9] proposed algorithm that considers the characteristics of different tasks and dynamically assigns them to appropriate cloud resources, resulting in improved performance and cost efficiency on hybrid cloud. There is a significant improvement in makespan and cost but load balance is not considered.

Jafar [10] proposed a hybrid optimization technique using GWO and Whale Optimization algorithms to schedule independent tasks to optimize cost, energy consumption and makespan. The makespan improvement is compared against GWO and WO. The algorithm is not compared against PSO and not tested on workflows.

HWACOA scheduler [11] applied Ant colony optimization with the concept of weight is employed to map the VMs with tasks to improve makespan and cost. The

A Adaptive PSO was proposed by [12] by considering heterogeneity in Cloud to schedule workflows to reduce the cost and makespan of execution of tasks. The authors employed adapative intertia weight to improve the global search.

Clustering based technique was employed by [13] to choose based on the execution time and availability of the resource to improve the cost of the workflow.

Kalka et. Al. (2021) [14] proposed multi-objective variant of particle swarm optimization (PSO) called Constriction Coefficient-based Multi-objective PSO (CCMOPSO) to solve the task scheduling problem. Constriction coefficient to control the velocity of the particles was used to improve the makespan and resource utilization.

Shahin et al. (2019) [15] proposed a workflow scheduling algorithm that uses a cuckoo search (CS) algorithm to optimize cost and load balance in cloud computing. They showed that the algorithm outperformed other algorithms in terms of makespan and load balance but did not consider cost.

Prerit chawda and Partha Sarathi [16] proposed a load balancing technique to increase resource utilization and minimize makespan using Min-Min algorithm for the independent task scheduling problem. The authors have not considered cost of the tasks and not implemented for workflow tasks.

The proposed study aims to contribute to this field by proposing a workflow scheduling algorithm that minimizes makespan, cost, and improves load balance on resources using PSO and GWO algorithms in cloud computing.

## III. PROPOSED METHODOLOGY

### *A. Problem Formulation*

The workflow scheduling problem can be represented mathematically as a mapping function [17].

*International Journal on Recent and Innovation Trends in Computing and Communication*
*ISSN: 2321-8169 Volume: 11 Issue: 9s*
*DOI: https://doi.org/10.17762/ijritcc.v11i9s.7744*
***Article Received**: 19 May 2023 **Revised:** 20 July 2023 **Accepted:** 07 August 2023*

$$M_f(\text{Resource}, T):\ T \rightarrow \text{Resource}$$

where Resource refers to set of virtual machines, and T is a set of tasks, each with its unique characteristics. Scheduling algorithms are designed using these characteristics to generate mappings that optimize the targeted objectives. The objectives considered are Makespan, Cost and Load balance.

Makespan refers to the total execution time to complete all the tasks from the start task to last task in a workflow given by eq. (1). Makespan helps to ensure timely completion of the entire workflow.

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

Cost of workflow execution given in eq. (c) is computed based on execution time of tasks run on VMs times the price of a VM and data transfer cost. Data transfer time is dependent on the bandwidth of the channel. Execution time of tasks is given in eq. (2).

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

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

Where
$e_{ti}$ refers to the execution time of task i
$P_{vmj}$ refers to the Price of a VM j ; $1 \leq j \leq Max.No.of\ VMs$
$dt_i$ refers to the size of the input data to task i, it is zero if input data is available on the VM on which ti is running.
$P_{BW}$ refers to Price for bandwidth

Load balance is one of the important parameters that ensures all the resources or VMs are utilized to the same extent and avoid overloading on a particular resource. We have applied the load balance rate as mentioned in eq. (d) as the measure to determine the load balance. The load balance rate refers to the measure of how evenly the workload is distributed across the available resources. A value of 1 (ideal scenario) indicates perfect load balance, meaning that all resources are being utilized equally. A value greater than 1 indicates extent of imbalance, with some resources being utilized more than others. Value nearer to 1 indicates a good balance.

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

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.

## B. *Proposed Algorithm*

The proposed approach employs GWO and proposed Constriction factor based PSO algorithm each for n/2 iterations respectively, where n is the total number of iterations.

Grey Wolf Optimization (GWO) is an optimization algorithm (Algorithm.1) that is inspired by the hunting behavior of grey wolves in the forest or nature. GWO algorithm was proposed by Mirjalili et al. in 2014. In GWO, a population of candidate solutions is represented by a pack of grey wolves. The algorithm imitates the social hierarchy and hunting behavior of the wolves to update the candidate solutions and find the optimal solution for a given optimization problem.

The process followed by the GWO algorithm are as follows:

1. Initialization: A population of candidate solutions (wolves) is randomly generated.
$$Solutions = (\overrightarrow{X_1}, \overrightarrow{X_2}, \overrightarrow{X_3} \dots \dots \overrightarrow{X_N})$$
2. Hunting behavior: The alpha, beta, and delta wolves the primary wolves that help or direct the other wolves in the pack. They are identified based on their fitness values, which are used to update the position of the other wolves. The alpha wolf has the highest fitness value, followed by the beta wolf and then the delta wolf. Once the alpha, beta, and delta wolves are identified, they are used to update the position of the other wolves in the pack. The position update equation for each wolf is given by eq. (5), (6) and (7). The new position is computed using the eq. (8). All the position, distance and coefficients represent a vector.

$$\overrightarrow{D_\propto} = \left|\overrightarrow{C_1} * \overrightarrow{X_\alpha} - \vec{X}\right|$$
$$\overrightarrow{X_1} = \overrightarrow{X_\propto} - \overrightarrow{A_1}.(\overrightarrow{D_\propto}) \quad (5)$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2} * \overrightarrow{X_\beta} - \vec{X}\right|$$
$$\overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.(\overrightarrow{D_\beta}) \quad (6)$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3} * \overrightarrow{X_\delta} - \vec{X}\right|$$
$$\overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.(\overrightarrow{D_\delta}) \quad (7)$$

$$\overrightarrow{X_{new}} = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \quad (8)$$

where X is the current position of the wolf, X_new is the updated position, A_alpha, A_beta, and A_delta are the positions of the alpha, beta, and delta wolves, respectively, and C1 and C2 are constants that control the exploration and exploitation capabilities of the algorithm.

3. Encircling behavior: The wolves try to encircle the prey (optimal solution) by updating their positions as shown in eq. (5).
4. Attacking behavior: The wolves try to attack the prey by updating their positions towards the optimal solution using eq. (9) and eq. (10)

$$\vec{A} = (2 \times \vec{a} \times \overrightarrow{r_1}) - \vec{a} \quad (9)$$
$$\vec{C} = (2 \times \overrightarrow{r_2}) \quad (10)$$

Where $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vectors with values ranging between 0 and 1.

*International Journal on Recent and Innovation Trends in Computing and Communication*
***ISSN: 2321-8169 Volume: 11 Issue: 9s***
***DOI: https://doi.org/10.17762/ijritcc.v11i9s.7744***
***Article Received**: 19 May 2023 **Revised:** 20 July 2023 **Accepted:** 07 August 2023*

5. Updating the population: The updated positions of the wolves are used to generate a new population of candidate solutions. The wolves with the best fitness values are kept as the new alpha, beta, and delta wolves, and the other wolves are randomly generated within the search space.
6. Termination: The algorithm terminates when the loop reaches the maximum number of iterations. The best solution found i.e. alpha wolf by the pack is returned as the final solution.

**Algorithm1.** Grey Wolf Optimization

**Input**: Workflow Objective function to optimize, population size (N), total or maximum number of iterations (max_iter), search space (bounds), and initial alpha, beta, and delta values

**Output**: Best solution found by the algorithm

```
// Initialize the population with random solutions
for i = 1 to N do
  solution_i (X⃗) = generate_random_solution(bounds)
  fitness_i = evaluate_fitness(solution_i)
  update_alpha_beta_delta(solution_i)
end for

for iter = 1 to max_iter/2 do
  // Update the positions of the wolves based on the alpha, beta, and delta values
  for i = 1 to N do
    a = 2 - 2 * iter / max_iter
    r1 = random_number(0, 1)
    r2 = random_number(0, 1)

    A = 2 * a * r1 - a    // Calculate the coefficient A
    C = 2 * r2            // Calculate the coefficient C

    X_alpha = alpha.position
    D_alpha = C * (X_alpha - solution_i.position)

    X_beta = beta.position
    D_beta = C * (X_beta - solution_i.position)

    X_delta = delta.position
    D_delta = C * (X_delta - solution_i.position)

    new_position = ((X_alpha - A * D_alpha) + (X_beta - A * D_beta) + (X_delta - A * D_delta))/3
    new_position = clip(new_position, bounds)

    new_fitness = evaluate_fitness(new_position)
    if new_fitness < fitness_i then
      solution_i.position = new_position
      fitness_i = new_fitness
      update_alpha_beta_delta(solution_i)
    end if
  end for
end for
// Return the best solution or alpha wolf position
return alpha
```

Particle Swarm Optimization (PSO) is a popular optimization algorithm that is inspired by the social behavior of swarms of birds or fish. The algorithm finds the optimal solution by iterating through the candidate solutions called particles of an optimization problem. At the start of the algorithm, the population of particles are initialized randomly within the search space of the optimization problem. Each particle represents a candidate solution, and its position in the search space is determined by a vector of decision variables. Each particle also has a velocity vector as mentioned in eq. (11) that determines its direction and speed of movement within the search space.

$$cognitive_{component} = c1 * r1 * \left(pbest(i) - solution(i)\right)$$

$$social\_component = c2 * r2 * (gbest - solution(i))$$

$$velocity\ (i+1) = \omega * velocity(i) + cognitive\_component + social\_component \tag{11}$$

In each iteration of the algorithm, the particles move through the search space according to two rules: personal best and global best. The personal best rule allows each particle to remember the best solution it has found so far, while the global best rule allows the particles to learn from the best solution found by the entire swarm. In the proposed algorithm for calculating the velocity vector a constriction factor-based inertia weight as mentioned in eq. (12) is introduced to improve exploratory capability of the algorithm.

$$\omega = \frac{\omega_{max} - \omega_{min}}{\varphi} + (\varphi\ (\omega_{n-1} - \omega_{max}) + \omega_{max}) \tag{12}$$

$\omega_{max}$ = Max. Inertia Weight Wight
$\omega_{min}$ = Minimum Inertia Weight
$\omega_{n-1}$ = Previous Inertia Weight
$\varphi$ = Constriction factor

The proposed constriction factor based PSO algorithm employs the constriction factor (φ) to update the inertia weight (ω). It is used to maintain the balance between exploration and exploitation in the algorithm. Clerc [18] in 1999 introduced Constriction factor in his study on convergence and stability of PSO. φ shown in eq. (13) is a parameter that restricts the velocity of particles during the PSO optimization process. The motive behind the constriction factor is to ensure that the particles converge to the global optimum in a stable and efficient manner, while also preventing them from overshooting the optimal solution.

$$\varphi = \frac{2}{\left|2-c_1-c_2-\sqrt{{c_1}^2+{c_2}^2+2c_1c_2-2c_1-2c_2+4}\right|} \tag{13}$$

where c_1and c_2 are the acceleration coefficients used in the velocity update equation and ensures the values is in the range [0,1].

**Algorithm2.** Proposed Constriction Factor based Particle Swarm Optimization

**Input**: Objective function to optimize, population size (N), maximum number of iterations (max_iter), search space (bounds), and constriction factor based inertia weight (w), cognitive parameter (c1), and social parameter (c2)

**Output**: Best solution found by the algorithm

```
1. Initialize the gbest with the alpha wolf returned by Algorithm1
2. Initialize the population with random solutions and velocities
   - For each particle i from 1 to N do the following:
     - Generate a random solution within the search space (bounds)
     - Generate a random velocity within the search space (bounds)
     - Evaluate the fitness of the particle's solution
     - Set pbest_i to the particle's solution
     - Set pbest_fitness_i to the particle's fitness
     - If pbest_fitness_i is less than gbest_fitness,
   set gbest to pbest_i and gbest_fitness to pbest_fitness_i
3. For iter from 1 to max_iter/2 do the following:
   - For each particle i from 1 to N do the following:
     - Generate two random numbers r1, r2 between 0 and 1
     - Compute cognitive and social components
     - Compute the new velocity of the particle
     - Compute the new position of the particle:
    - Clip the new_position_i to be within the search
       space (bounds)
     - Evaluate the fitness of the new solution
       - If the new fitness is better than pbest_fitness_i,
         update pbest_i and pbest_fitness_i
       - If pbest_fitness_i is less than gbest_fitness,
          set gbest to pbest_i and gbest_fitness
          to pbest_fitness_i
     - Update the particle's velocity and solution with
        the new values

4. Return the best solution found by the algorithm (gbest)
```

The proposed Hybrid GWO-CPSO Optimization algorithm (HGWOCPSO) is designed to schedule the workflows on cloud to minimize makespan, cost and maximize load balance. Firstly, the GWO algorithm is applied to promote exploration and later PSO algorithm to promote exploitation. GWO is initialized with random particles (the population). Particles represent the workflow solutions generated randomly. These solutions are run through GWO algorithm for 50% of the maximum iterations considered and then PSO is run for the remaining 50% of iterations.

**Algorithm 3.** Proposed Hybrid GWO-CPSO Optimization

**Input**: Workflow tasks, cloud resources, number of iterations (n), population size (p), and other parameters of GWO and CPSO

**Output**: Solution that has best makespan, cost, and load rate values obtained by the hybrid algorithm

```
// Initialization
Initialize the GWO and PSO populations randomly
Evaluate the fitness of each wolf and particle in the population
Set the global best position and fitness for PSO

// Main loop
for i = 1 to n/2 do
  if i <= n/2 then
    // GWO iterations
    for j = 1 to p do
      Update the positions of the wolves using
      GWO operators
      Evaluate the fitness of each wolf in the population
      Update the alpha, beta, and delta wolves
    end for
  for i = n/2 to n do
    // PSO iterations
    for j = 1 to p do
      Update the positions and velocities of the particles
       for PSO
      Evaluate the fitness of each particle in the population
      Update the local and global best positions and fitness
    end for
  end if

  // Update the best position and fitness values
  Update the global best position and fitness for PSO
  Update the best makespan, cost, and resource utilization values
obtained so far

end for

return the solution that has optimized makespan, cost, and load balance
values
```

## IV. PERFORMANCE EVALUATION

This section presents evaluation of proposed HGWOCPSO algorithm by performing extensive simulation experiments with three types of scientific workflows.

### *A. Experimental Setup*

The proposed Hybrid GWO-PSO Optimization algorithm (HGWOCPSO) The proposed algorithm is implemented on Workflowsim [19] based on cloudsim and run on four types of scientific workflows Montage, Epigenomics, and Sipht

[20–23]. These workflows are available with 30, 50, 100 and 1000 tasks. Experiments were conducted with Workflowsim on Intel i5 10th gen CPU, 8GB RAM, Windows 10 64-bit OS. The experimentation utilizes heterogenous VMs and variable costs for VM instances.

The Fitness function shown in eq. (14) is adopted to optimize the three parameters makespan, cost and load balance is represented in eq.

$$Fitness = \alpha \times \left(\frac{1}{Makespan}\right) + \beta \times \left(\frac{1}{Cost}\right) - \gamma \times \left(\frac{1}{Load\ balance\ rate}\right) \quad (14)$$

α, β, γ represent the weights assigned to each parameter. Here α=0.4, β=0.3 and γ=0.3 are considered

The parameters considered for simulation of the algorithms are represented in Table 1.

TABLE I. PARAMETERS CONSIDERED FOR ALGORITHM

| ***Parameters*** | ***Values*** |
|---|---|
| Number of Tasks | 25-1000 |
| Number of particles | 100 |
| Number of iterations (Max_iter) | 500 |
| r1, r2 | random(0,1) |
| c1,c2 | 2 |
| w | constriction factor |
| Number of VMs | 5 (heterogenous) |
| Bandwidth | 100 |

## B. *Performance Analysis*

The results of the proposed algorithm HGWOCPSO are compared with Particle Swarm Optimization (PSO), Grey Wolf Optimization (GWO), Chaotic PSO (CPSO), hybrid GWO-PSO algorithm (GWOPSO). The comparison is made in terms of Makespan, Cost and Load rate for the three scientific workflows of different sizes and are presented in Table 2 and Table 3.

TABLE II. MAKESPAN FOR WORKFLOWS

| **Dataset** | ***GWO*** | ***PSO*** | ***CPSO*** | ***GWOPSO*** | ***HGWOCPSO*** |
|---|---|---|---|---|---|
| Montage 25 | 182.7 | 159.88 | 158.5 | 156.16 | 146.97 |
| Montage 50 | 407.7 | 245.92 | 284.79 | 279.37 | 277.5 |
| Montage 100 | 864.9 | 623.5 | 484.41 | 494 | 491.12 |
| Montage 1000 | 9117.0 | 5679.97 | 5663.61 | 5642.9 | 5415.04 |
| Epigenomics_24 | 14235.2 | 15147.16 | 9882.78 | 8835.23 | 7339.48 |
| Epigenomics_46 | 33184.2 | 28551.56 | 22125.68 | 23912.97 | 17638.59 |
| Epigenomics_100 | 322937.0 | 154742 | 153158.3 | 160209.2 | 154308 |
| Epigenomics_997 | 2084102.8 | 1513756 | 1500285 | 1472362 | 1344813 |
| Sipht_30 | 6664.4 | 6130.58 | 6065.01 | 4464.64 | 4213 |
| Sipht_60 | 9341.1 | 10259.01 | 7408.61 | 10468.14 | 9005.57 |
| Sipht_100 | 13910.1 | 10967.75 | 10237.62 | 10503.37 | 9266.29 |
| Sipht_1000 | 104213.1 | 65610.42 | 68068.19 | 64319.2 | 50756.48 |

TABLE III. AVERAGE MAKESPAN FOR WORKFLOWS (WF)

| **Workflows** | ***GWO*** | ***PSO*** | ***CPSO*** | ***GWOPSO*** | ***HGWOCPSO*** |
|---|---|---|---|---|---|
| Montage | 2643.1 | 1677.3175 | 1647.8275 | 1643.1075 | 1582.6575 |
| Epigenomics | 613614.8 | 428049.18 | 421362.94 | 416329.85 | 378524.76 |
| Sipht | 33532.1585 | 23241.94 | 22944.8575 | 22438.83 | 18310.335 |

The Average Makespan of the proposed HGWOCPSO algorithm shows 4% increase over CPSO & GWOPSO, 6% increase over PSO and 67% over GWO for Montage workflows. It shows 10%, 11%, 13%, 62% increase over GWOPSO, CPSO, PSO and GWO for Epigenomics workflows respectively. Also it shows 22%, 25%,27% and 83% improvement over GWOPSO, CPSO, PSO and GWO for Sipht workflows respectively.
Makespan improvement is higher for Sipht and Epigenomics workflows. Figure 2, Figure 3 & Figure 4 shown below indicate the significant performance improvement by proposed HGWOCPSO algorithm for makespan.

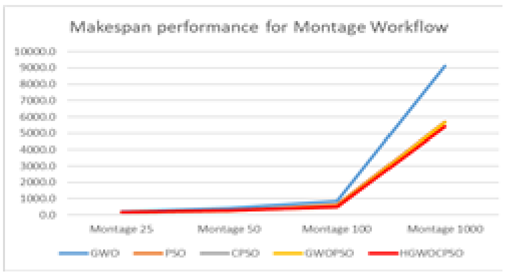

Figure 2. Makespan improvement for Montage Workflow

*International Journal on Recent and Innovation Trends in Computing and Communication*
***ISSN: 2321-8169 Volume: 11 Issue: 9s***
***DOI: https://doi.org/10.17762/ijritcc.v11i9s.7744***
***Article Received**: 19 May 2023 **Revised:** 20 July 2023 **Accepted:** 07 August 2023*

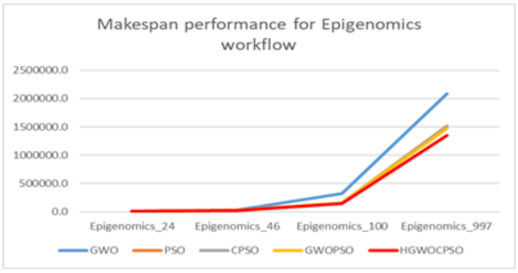

Figure 3. Makespan improvement for Epigenomics Workflow

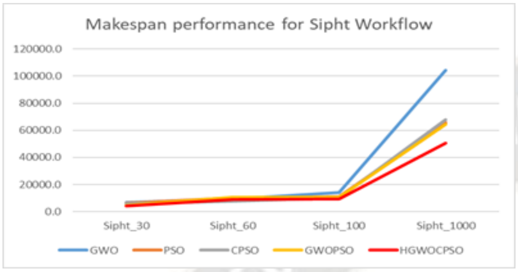

Figure 4. Makespan improvement for Sipht Workflow

The improvement in terms of cost parameter is shown in Table 4 and Table 5 respectively for the workflows.

TABLE IV. COST OF WORKFLOWS

| **Dataset** | ***GWO*** | ***PSO*** | ***CPSO*** | ***GWOPSO*** | ***HGWOCPSO*** |
|---|---|---|---|---|---|
| Montage 25 | 2107.66 | 870.15 | 926.27 | 900.29 | 831.1 |
| Montage 50 | 4686.31 | 1938.31 | 1842.33 | 1786.56 | 1742.1 |
| Montage 100 | 9917.37 | 4114.23 | 3996.3 | 3955.94 | 4105.8 |
| Montage 1000 | 104410.97 | 44694.06 | 44575.9 | 44355.76 | 42050.7 |
| Epigenomics_24 | 161447.05 | 57887.29 | 50400.24 | 48811.84 | 45185.1 |
| Epigenomics_46 | 375392.55 | 144491.48 | 146306.36 | 140673.2 | 122315.5 |
| Epigenomics_100 | 3647501.69 | 1384239.84 | 1436833.7 | 1306006 | 1200988.6 |
| Epigenomics_997 | 34829492.34 | 13737618.67 | 13833360 | 13202032 | 13165323.8 |
| Sipht_30 | 50008.91 | 18496.56 | 18336.11 | 15168.42 | 18281.8 |
| Sipht_60 | 105132.18 | 43675.8 | 43151.55 | 43829.66 | 30970.7 |
| Sipht_100 | 156557.46 | 56230.03 | 58303.69 | 55079.54 | 56697.5 |
| Sipht_1000 | 1563797.03 | 606803.62 | 598374.4 | 587908.4 | 585141 |

TABLE V. AVERAGE COST OF WORKFLOWS (WF)

| **Workflows** | ***GWO*** | ***PSO*** | ***CPSO*** | ***GWOPSO*** | ***HGWOCPSO*** |
|---|---|---|---|---|---|
| Montage | 30280.6 | 12904.19 | 12835.2 | 12749.64 | 12182.41 |
| Epigenomics | 9753458.4 | 3831059 | 3866725 | 3674381 | 3633453 |
| Sipht | 33532.1585 | 23241.94 | 22944.8575 | 22438.8375 | 18310.335 |

The Average Cost of the proposed HGWOCPSO algorithm shows 5%, 2%, 2% increase over GWOPSO , CPSO & GWO for Montage, Epigenomics and Sipht workflows. There is a significant cost improvement for Montage and Sipht workflows.
The cost of executing all the three workflows on HGWOCPSO algorithm indicates improvement over other algorithms. The percentage improvement is though minimal the load rate and makespan show significant improvement. Figure 5, Figure 6 and Figure 7 show the Cost of execution of three workflow types.

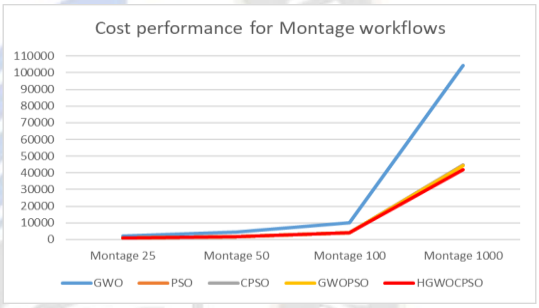

Figure 5: Cost improvement for Montage Workflows

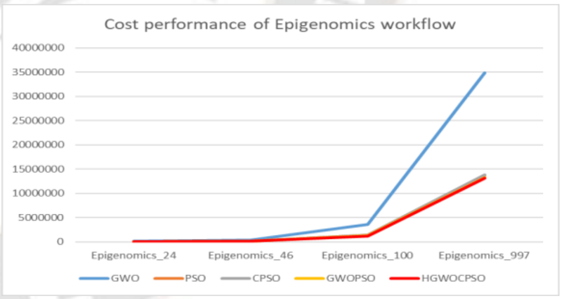

Figure 6: Cost improvement for Epigenomics Workflows

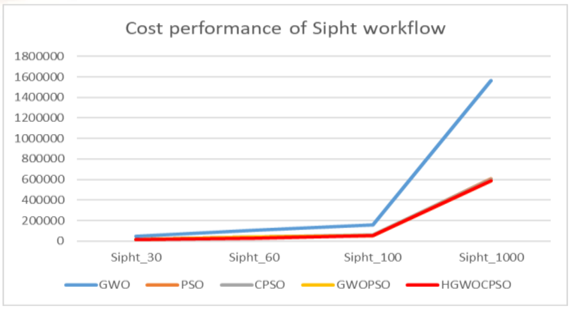

Figure 7: Cost improvement for Sipht Workflows

***International Journal on Recent and Innovation Trends in Computing and Communication***
***ISSN: 2321-8169 Volume: 11 Issue: 9s***
***DOI: https://doi.org/10.17762/ijritcc.v11i9s.7744***
***Article Received**: 19 May 2023 **Revised:** 20 July 2023 **Accepted:** 07 August 2023*

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.

TABLE VI. LOADBALANCE OF WORKFLOWS

| **Dataset** | ***GWO*** | ***PSO*** | ***CPSO*** | ***GWOPSO*** | ***HGWOCPSO*** |
|---|---|---|---|---|---|
| Montage 25 | 1.86 | 1.22 | 1.25 | 1.23 | 1.12 |
| Montage 50 | 1.78 | 1.16 | 1.15 | 1.14 | 1.08 |
| Montage 100 | 1.78 | 1.16 | 1.15 | 1.14 | 1.12 |
| Montage 1000 | 1.78 | 1.16 | 1.15 | 1.14 | 1.08 |
| Epigenomics_24 | 1.57 | 1.14 | 1.15 | 1.11 | 1.12 |
| Epigenomics_46 | 1.57 | 1.15 | 1.14 | 1.11 | 1.09 |
| Epigenomics_100 | 1.57 | 1.14 | 1.13 | 1.12 | 1.09 |
| Epigenomics_997 | 1.57 | 1.11 | 1.11 | 1.11 | 1.09 |
| Sipht_30 | 1.4 | 1.11 | 1.13 | 1.14 | 1.08 |
| Sipht_60 | 1.4 | 1.11 | 1.13 | 1.14 | 1.09 |
| Sipht_100 | 1.4 | 1.11 | 1.13 | 1.09 | 1.08 |
| Sipht_1000 | 1.4 | 1.11 | 1.13 | 1.09 | 1.09 |

TABLE VII. AVERAGE LOADBALANCE OF WORKFLOWS (WF)

| **Workflows** | ***GWO*** | ***PSO*** | ***CPSO*** | ***GWOPSO*** | ***HGWOCPSO*** |
|---|---|---|---|---|---|
| Montage | 1.8 | 1.17 | 1.17 | 1.16 | 1.1 |
| Epigenomics | 1.6 | 1.13 | 1.13 | 1.11 | 1.09 |
| Sipht | 1.4 | 1.11 | 1.13 | 1.11 | 1.08 |

The result analysis shows that the overall trend of makespan and load balance is higher for workflows with large number of tasks and there is an improvement in cost of execution of workflows.

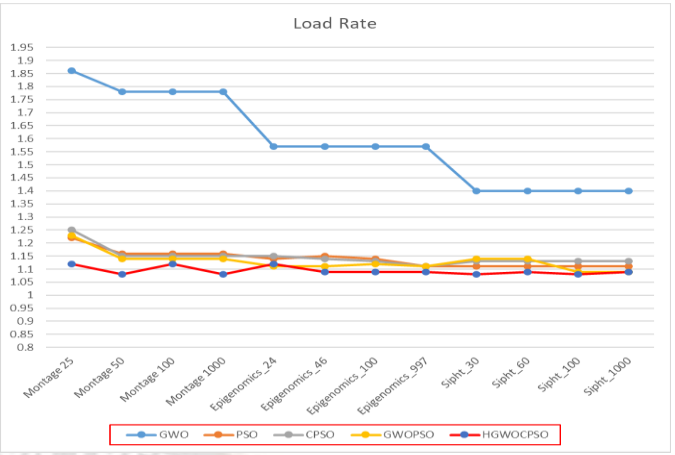

Figure 8: Load balance on resources for workflow execution on existing and proposed algorithms

## V. CONCLUSION AND FUTURE WORK

In conclusion, we proposed a constriction factor inertia weight-based Particle Swarm optimization (CPSO) and hybrid algorithm that employs both GWO and CPSO to optimize makespan, cost, and maximize load balance in scheduling workflows. The hybrid algorithm leverages the strengths of both algorithms and balances exploration and exploitation of the solution space to find near-optimal solutions.

The simulation results show that the proposed hybrid algorithm achieves better performance compared to GWO, PSO & Chaotic PSO algorithms alone and hybrid GWO-PSO algorithms in optimizing makespan, cost, and load balance. The algorithm also demonstrates good convergence and stability in the experiments, indicating its robustness and effectiveness in providing effective solution to the workflow scheduling problem in cloud computing.

In future research the work can be extended towards further improving the algorithm's performance by incorporating other optimization algorithms, considering more complex constraints, apply machine learning techniques along with hybrid optimization methods. Overall, the proposed hybrid GWOCPSO algorithm provides a promising approach to optimize workflow scheduling in cloud computing while considering multiple objectives.

## CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

## REFERENCES

[1] Mainak Adhikari , T Amgoth , Satish Narayana Srirama, “A Survey on Scheduling Strategies for Workflows in Cloud Environment and Emerging Trends”, ACM Computing Surveys 52 (4), pp1-36, 2019.
[2] Kennedy, J. and Eberhart, R.C., "Particle Swarm Optimization", Proceedings of the IEEE International Conference on Neural Networks, 1995.

***International Journal on Recent and Innovation Trends in Computing and Communication***
***ISSN: 2321-8169 Volume: 11 Issue: 9s***
***DOI: https://doi.org/10.17762/ijritcc.v11i9s.7744***
***Article Received**: 19 May 2023 **Revised:** 20 July 2023 **Accepted:** 07 August 2023*

[3] Mirjalili, S., S.M. Mirjalili, and Lewis, A., "Grey Wolf Optimizer", Advances in Engineering Software, 2014.

[4] A. Maria, Rodriguez, Rajkumar Buyya, “Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds” IEEE Transactions on Cloud Computing, pp. 222-235, 2014.

[5] Chaudhary, A. ., Sharma, A. ., & Gupta, N. . (2023). A Novel Approach to Blockchain and Deep Learning in the field of Steganography. International Journal of Intelligent Systems and Applications in Engineering, 11(2s), 104–115. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/2514

[6] Chung-Feng Wang, Kui Liu, "A Novel Particle Swarm Optimization Algorithm for Global Optimization", Computational Intelligence and Neuroscience, vol. 2016.

[7] Jian Chang, Zhigang Hu, Yong Tao, Zhou, “Task Scheduling Based on Dynamic Non-Linear PSO in Cloud Environment” IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018.

[8] Dwarkanath Pande, S. ., & Hasane Ahammad, D. S. . (2022). Cognitive Computing-Based Network Access Control System in Secure Physical Layer. Research Journal of Computer Systems and Engineering, 3(1), 14–20. Retrieved from https://technicaljournals.org/RJCSE/index.php/journal/article/view/36

[9] Bilal Abed-alguni, Noor Aldeen Alawad, "Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments", Applied Soft Computing, 2021.

[10] Abdullah Alzaqebah, Rizik Al-Sayyed, Raja Masadeh, "Task Scheduling based on Modified Grey Wolf Optimizer in Cloud Computing Environment", 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019.

[11] Abdullah Alzaqebah, Rizik Al-Sayyed, Raja Masadeh, "Task Scheduling based on Modified Grey Wolf Optimizer in Cloud Computing Environment", 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019.

[12] Jafar Ababneh, "A Hybrid Approach Based on Grey Wolf and Whale Optimization Algorithms for Solving Cloud Task Scheduling Problem", Journal of Mathematical Problems in Engineering, 2021.

[13] Chirag Chandrashekar, Pradeep Krishnadoss, Vijaya kumar Kedalu Poornachary, Balasundaram Ananthakrishnan, and Kumar Rangasamy , "HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing", Cyber–Physical Systems in Real-Time and Edge Computing for Smart Grids, 2023.

[14] Mwangi, J., Cohen, D., Silva, C., Min-ji, K., & Suzuki, H. Feature Extraction Techniques for Natural Language Processing Tasks. Kuwait Journal of Machine Learning, 1(3). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/137

[15] Pengze Guo, Zhi Xue, "An adaptive PSO-based real-time workflow scheduling algorithm in cloud systems", International Conference on Communication Technology (ICCT), 2017.

[16] D.A.Prathibha, B.Latha, and G. Sumathi, "Efficient scheduling of workflow in cloud enviornment using billing model aware task clustering", Journal of Theoretical and Applied Information Technology, 2014.

[17] Kalka Dubey, S.C. Sharma, “A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing”, Sustainable Computing: Informatics and Systems, Volume 32, 2021, 100605, ISSN 2210-5379.

[18] Shahin Ghasemi, Ali Hanani,"A Cuckoo-based Workflow Scheduling Algorithm to Reduce Cost and Increase Load Balance in the Cloud Environment", International Journal on Informatics Visualization, Vol.3, 2019.

[19] Prof. Virendra Umale. (2020). Design and Analysis of Low Power Dual Edge Triggered Mechanism Flip-Flop Employing Power Gating Methodology. International Journal of New Practices in Management and Engineering, 6(01), 26 - 31. https://doi.org/10.17762/ijnpme.v6i01.53

[20] Prerit Chawda, Partha Sarathi Chakraborty, "An Improved Min-Min Task Scheduling Algorithm for Load Balancing in Cloud Computing", ", International Journal on Recent and Innovation Trends in Computing and Communication, 2016.

[21] 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.

[22] Mei Chen, Machine Learning for Energy Optimization in Smart Grids , Machine Learning Applications Conference Proceedings, Vol 2 2022.

[23] Clerc M., The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, Proceedings of the 1999 Congress on Evolutionary Computation, 1999, Vol. 3, pp. 1951-1957.

[24] Chen, Weiwei & Deelman, Ewa., “WorkflowSim: A toolkit for simulating scientific workflows in distributed environments”, IEEE 8th International Conference on E-Science, e-Science 2012.

[25] Singh V, Gupta I, Jana PK, "An energy efficient algorithm for workflow scheduling in iaas cloud", J Grid Comput:1–20, 2019.

[26] Gao Y, Zhang S, Zhou J, "A hybrid algorithm for multi-objective scientific workflow scheduling in iaas cloud", IEEE Access 7:125783–125795, 2019.

[27] Dubey K, Shams MY, Sharma S, Alarifi A, Amoon M, Nasr AA, "A management system for servicing multi-organizations on community cloud model in secure cloud environment", 2019, IEEE Access 7:159535–159546.

[28] Xie Y, Zhu Y, Wang Y, Cheng Y, Xu R, Sani AS, Yuan D, Yang Y, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment. Futur Generation Computer Systems 97:361–378", 2019.