HPSOGWO: A Hybrid Algorithm for Scientific Workflow Scheduling in Cloud Computing

Neeraj Arora[1]
School of Science and Technology
Vardhman Mahaveer Open University
Kota, Rajasthan, India 324010

Rohitash Kumar Banyal[2]
Department of Computer Science and Engineering
Rajasthan Technical University
Kota, Rajasthan, India 324010

***Abstract*—Virtualization is one of the key features of cloud computing, where the physical machines are virtually divided into several virtual machines in the cloud. The user's tasks are run on these virtual resources as per the requirements. When the user requests the services to the cloud, the user's tasks are allotted to the virtual resources depending on their needs. An efficient scheduling mechanism is required for optimizing the involved parameters. Scientific workflows deals with a large amount of data with dependency constraints and is used to simplify the applications in the diverse scientific domains. Scheduling the workflow in cloud computing is a well-known NP-hard problem. Deploying such data- and compute-intensive workflow on the cloud needs an efficient scheduling algorithm. In this paper, we have proposed a multi-objective model based hybrid algorithm (HPSOGWO), which combines the desirable characteristics of two well-known algorithms, particle swarm optimization (PSO), and grey wolf optimization (GWO). The results are analyzed under complex real-world scientific workflows such as Montage, CyberShake, Inspiral, and Sipht. We have considered the two essential parameters: total execution time and total execution cost while working in the cloud environment. The simulation results show that the proposed algorithm performs well compared to other state-of-the-art algorithms such as round-robin (RR), ant colony optimization (ACO), heterogeneous earliest time first (HEFT), and particle swarm optimization (PSO).**

Keywords*—Cloud computing; hybrid algorithms; metaheuristic algorithms; optimization; workflow scheduling

I. Introduction

Cloud Computing is a buzzing word from decades in computer science as it offers advancements like hiding and abstraction of complexity, visualized resources, and efficient use of distributed resources. Few well-known cloud computing platforms are Amazon EC2, GoGrid, Google App Engine, Microsoft Azure, etc. [1] [2]. The services of cloud computing can be classified into Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [3].

The tasks are dependent on each other in the workflow. The workflow can be represented using Directed Acyclic Graph (DAG) [4], where the nodes in the DAG represent the tasks (T), and the edges (E) joining the nodes represent the dependency between the tasks. A sample workflow is shown in Fig. 1, containing eight tasks $\{T1, T2, T3, T4, T5, T6, T7, T8\}$. The tasks $T1$ and $\{T4, T6, T7, T8\}$ are the entry and exit tasks, respectively. Each edge of the DAG shows the dependencies between the tasks. For example, $T2$ is executed after $T1$ which is shown by the paired set $\{T1, T2\}$. A scientific workflow is a specialized form of the workflow which is used in various scientific domains like astrology, bio-informatics, gravitational waves, etc. [5].

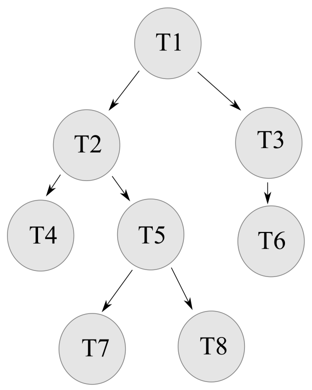


Fig. 1. Dependency of Tasks for Workflow Schedule.

Pegasus project published some of the realistic scientific workflows like Montage, CyberShake, Epigenomics, LIGO, and SIPHT [6] [7]. The structures of these workflows are shown in Fig. 2.

The workflow scheduling can be considered as a mapping function where several dependent tasks are mapped to several available virtual machines [9]. Suppose, m number of tasks maps to the n number of virtual machines; then, n^m combinations which are possible if the brute force algorithm is used. So, the workflow scheduling is a complex problem, and the solution is not found in polynomial time [10]. It is good to find a near-optimal solution to the workflow scheduling problem with a meta-heuristic algorithm.

Many meta-heuristic optimization algorithms have been used to solve workflow problems in cloud computing. Genetic Algorithm (GA) is used in workflow scheduling to minimize the makespan [11]. GA algorithm is robust and generates a high-quality search in polynomial time but takes a bit more time to find the solution. Pandey et al. [12] used Particle Swarm Optimization (PSO) to schedule workflow applications in a cloud computing environment. PSO is a fast optimization algorithm but has a problem such as earlier convergence and trapping in local optimal solution [13]. Grey Wolf Optimization (GWO) is the recent proposed meta-heuristic algorithm that mimics grey wolves' leadership hierarchy [14]. Khalil and Babamir [15] offered the extended version of Grey Wolf Optimizer for solving the workflow problem. GWO reduces the

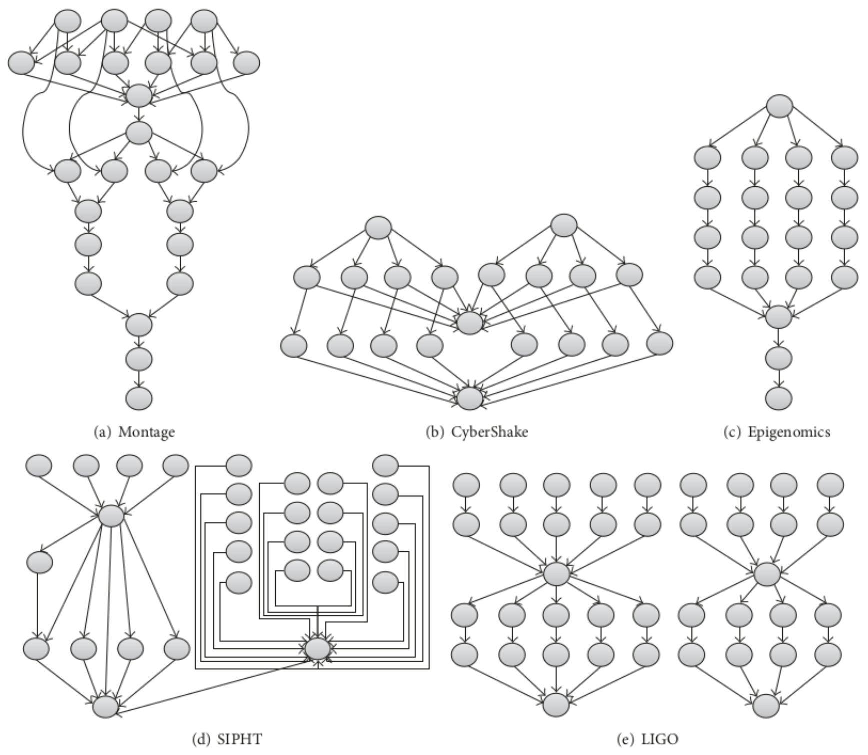


Fig. 2. Structures of Real-World Scientific Workflows [8].

probability of being trapped in the local optimal solution. By combining two or more algorithms considering their strengths, one can overcome the aforementioned issues of algorithms. In this paper, we proposed a hybrid algorithm combining Particle Swarm Optimization (PSO) and Grey Wolf Optimize (GWO), named HPSOGWO. The HPSOGWO is tested on the scientific workflow like montage, cybershake, inspiral, and sipht to optimize total execution cost and time. In the next section, we review some of the scheduling algorithms used in cloud computing.

II. Related Work

Workflow is more popular among the scientists in which a complex scientific process is modeled into small tasks [16]. These tasks can be executed on parallel and distributed computing like cloud computing. Workflow scheduling is a well-known NP-hard problem in cloud computing. Several list based heuristics have been proposed for task scheduling to optimize the performance of cloud computing like first come first serve (FCFS), round-robin (RR), shortest job first (SJF), minimum completion time (MCT), etc. The basic idea of list-based heuristics is to assign a priority to each task and allot to the available resources as per given preferences. The Heterogeneous Earliest Finish Time (HEFT) was designed for heterogeneous multiprocessor systems. Dubey et al. [17] proposed a modified version of HEFT, capable of reducing the makespan time compared to existing HEFT and Critical Path on a Processor (CPOP).

In the Min-Min algorithm, the minimum execution time task is mapped to the machine with minimum completion time [18]. A similar algorithm is Max-Min algorithm where in the task with maximum execution time is assigned to the machine, which takes minimum completion time. The Min-Min and Max-Min are offline scheduling that work in batch mode, which means the tasks are not allocated to the resources as they enter [19]. Suffering from starvation is the drawback with Min-Min and Max-Min algorithms [20]. Besides, they consider only the time as a resource quality. The list-based heuristics concentrate only on the user perspectives; they are less focused on the resource quality parameters.

These aforementioned conventional heuristics algorithms are simple, easy to implement, and fast, but for further improvement in the quality of solution and to achieve the optimum results for complex problems like workflow scheduling, the meta-heuristic approaches can find the near-optimal solution [21]. In addition, the heuristic algorithms are problem-dependent techniques, whereas the meta-heuristic methods are problem-independent techniques. The meta-heuristic algorithms have been widely used due to its simplicity and strong searching power in less time and cost. Many of the meta-heuristics approaches were proposed for solving the workflow problem. Some of the standard algorithms are the Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO).

Dasgupta et al. [1] proposed a genetic-based algorithm used for task scheduling problem. The experiment results show better performance in terms of makespan when compared with First Come First Serve (FCFS), Round Robing (RR), and a local search algorithm Stochastic Hill Climbing (SHC). The GA algorithm was reported to be a time-consuming algorithm for reaching the optimum solutions [22].

Tawfeek et al. [23] used the Ant Colony Optimization (ACO) approach for task scheduling to minimize the makespan and found that ACO performed better than FCFS and RR. However, ACO is a very complex algorithm and takes a long time to get optimal results [24]. The dependency of tasks is not been considered.

Particle swarm optimization (PSO) is one of the popular meta-heuristic algorithms. It is simple to implement and has fast convergences. Despite its advantages, it gets trapped in local optimum for the complex problems[25].

Meta-heuristic algorithms are characterized by exploitation and exploration abilities [26]. Exploitation means that the algorithm is very successful in performing local searches. Exploring means that the algorithm is useful to find out the initial solution, which may be near to the global optimum. A good meta-heuristic algorithm balances the exploration and exploitation abilities. Particle Swarm Optimisation has high exploration ability but is low in exploitation ability. The grey wolf optimizer is proposed by Mirjalili et al. [14] and has a right balance between exploration and exploitation abilities.

A single meta-heuristic might not get the optimal solution and may stuck into the local optimal solution for complex problems like scientific workflow scheduling. It is a better approach to combine one or more meta-heuristic algorithms based on their best characteristics. In the last few decades, hybrid algorithms have become popular. Here, we discuss only those existing algorithms which are either hybrid with PSO or GWO. Manasrah and Ali [8] proposed a hybridization of the Genetic Algorithm and Particle Swarm Optimization (GA-PSO) algorithm. The hybrid GA-PSO algorithm reduces the total execution time compared with GA, PSO, and other algorithms. Another hybrid algorithm has been reported in [27], which was the hybrid version of the PSO and gravitation search algorithm (GSA). This hybrid algorithm performs well compared with some non-heuristics, PSO, and GSA algorithms in terms of cost. The hybridization of the Grey Wolf Optimization (GWO) and Genetic Algorithm (GA) was proposed by Bouzary and Frank [28] and they found that the proposed algorithm was superior than GWO and genetic algorithm (GA) cost wise. Khurana and Singh [29] have introduced a hybrid of flower pollination algorithm and GWO to reduce the cost and time and give efficient results compared to flower pollination with genetic algorithm.

Despite the advantages of the aforementioned hybrid algorithms, one might ask the motivation behind the proposed algorithm. The answer lies in the free lunch theorem [30]. The free lunch theorem specifies that the single algorithm is not fit for solving all the optimization problems. It might perform better for a particular optimization problem, but it may not perform well for the other optimization problems. There is no universal solution to optimization problems.

III. Background

In this section, we discuss the standard PSO and GWO algorithm used in the designing the proposed algorithm. The fitness function used in the proposed algorithm is also explained in this section.

A. Fitness Function

The fitness function described the targeted objectives to be optimized using the proposed scheduling algorithm [31]. There are two approaches to make a fitness function multi-objective: priori and posteriori [32]. In the priori approach, each involved objective is assigned a weight, as per their importance, to make a single-valued function, also known as fitness value. Whereas, the set of non-dominant solutions is found in the posteriori approach. Here, we follow the priori approach to design the fitness function. The fitness function is the composition of total execution cost (TEC) and total execution time (TET). Mathematically, the considered fitness function can be represented using the equation 1.

$$f(TET, TEC) = \alpha_1 \times TET + \alpha_2 \times TEC \tag{1}$$

where TET and TEC are the total execution time (makespan) and total execution cost respectively. α_1 and α_2 are the weight assigned for each objectives. Here, we consider similar weight to α_1 and α_2 that is 0.5. The complete description of total execution time (makespan) and total execution cost is explained in the following sub-sections:-

1) Total execution time (makespan): The total execution time (makespan) is the maximum completion time taken by tasks in the workflow. In other words, makespan is the time required for finishing all the tasks allotted to different virtual machines [25]. Mathematically, the makespan of the workflow can be derived using equation 2.

$$TET_W = max\{CT_i \mid i = 1, 2, ...m\} \tag{2}$$

where CT_i is the completion time of the task T_i in the workflow. The completion time is the total execution time of the tasks. In case tasks are dependent, then the waiting time of predecessor tasks is also considered. The completion time CT_i is depicted in equation 3.

$$CT_i = \begin{cases} ET_i & \text{iff } pred(T_i) = \emptyset \\ WK_i + ET_i & \text{iff } pred(T_i) \neq \emptyset \end{cases} \tag{3}$$

The waiting time of task T_i is the maximum completion time of all the predecessor tasks of workflow as shown in equation 4.

$$WK_i = \begin{cases} 0 & \text{iff } pred(T_i) = \emptyset \\ max(CT_i) & \text{iff } pred(T_i) \neq \emptyset \end{cases} \tag{4}$$

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

The execution time of the task T_i on virtual machine VM_j is calculated using equation 5, where SZ_{Task} is the size of task T_i in million instruction (MI), $Num(PE_j)$ is the number of core assigned to the virtual machine VM_j, PE_{Unit} is the size of each core in MIPS.

2) Total execution cost: The cost is a prominent objective to be optimized as cloud computing follows a pay-as-you-go billing scheme [33]. Major of the cloud service providers charges for some specific time interval based on the cloud services used. Cost in cloud computing involves execution cost, communication cost, and storage cost. The total execution cost of VM is the cost charged of VM per unit interval and the execution time of tasks on that VM. Mathematically, the total execution cost (TEC) of workflow W is shown in equation 6 [34].

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

where CO_j is the cost of type-i VM instance for a unit time in the cloud data center. τ is the time period for which the resources are used by the user. $ET_{i,j}$ is the execution time of task T_i by type-j VM instance .

B. Particle Swarm Algorithm

Kennelly and Eberhat proposed the Particle Swarm Optimization (PSO) technique in 1995 [35]. It is a meta-heuristic technique based on the social behavior of the swarm of birds or particles. Each particle represents a solution for the problem and searches the optimal solution in the problem space. The particle is characterized by its position and velocity. In every iteration, the position and velocity are updated and moves towards the optimal results. PSO consists of the following stages:-

1) Evolve $gbest$ and $pbest$ of the Particles: In Particle Swarm Optimisation, each particle represents a solution and in each generation of particle it produces the global best particle denoted by $gbest$ and the personal best particle is denoted by $pbest$. The selection of $pbest$ and $gbest$ particles are determined by their fitness values.

2) Update Position and Velocity: Position and velocity of the particle are influenced by the personal best ($pbest$) and the global best particle ($gbest$).

$$\begin{aligned} V_i(t+1) = w.V_i(t) &+ C_1.r_1 * (pbest - x_i(t)) \\ &+ C_2.r_2 * (gbest - x_i(t)) \end{aligned} \tag{7}$$

Equation 7 represents the velocity of the i^{th} particle at the t iteration. The C_1 and C_2 are coefficient and w is the inertia weight. The initial values of the coefficient are given in Table II. r_1 and r_2 are the random numbers between 0 and 1.

$$x_i(t+1) = x_i(t) + V_i(t) \tag{8}$$

The position x of the i^{th} particle for t^{th} iteration is updated as the equation 8.

C. Grey Wolf Algorithm

Mirjalili et al. [14] proposed Grey Wolf Optimization (GWO) technique which mimics the hunting behaviour and leadership hierarchy of grey wolf. According to the mathematical model of the GWO, there are four types of wolves that are alpha (α), beta (β), delta (δ) and omega (Ω). Each wolf represents a solution. The alpha wolf represents the best solution. The second best solution and the third best solutions are represented by beta and delta wolves respectively and all the other rest solutions are known as omega wolf. GWO algorithm is composed of steps shown in Sections III-C1, III-C2, and III-C3.

1) Encircling prey: The grey wolf encircle prey during the process and this can be mathematically modelled using the equations 9 and 10.

$$D = |C.X_p(t) - X(t)| \tag{9}$$

$$X(t+1) = X_p(t) - A.D \tag{10}$$

The position of the wolf is updated using equations 9 and 10 for the current iteration t, where X_p is the position of prey and X is the position of wolf. A and C are the coefficient vectors and calculated using equations 11 and 12 respectively.

$$A = 2a.r_1 - a \tag{11}$$

$$C = 2.r_2 \tag{12}$$

The values in the random numbers r_1 and r_2 in equations 11 and 12 are in the range from 0 to 1 and the value of variable a is linearly decrease from 2 to 0 and is calculated using equation 16

2) Haunting: The alpha wolf (best solution) guides the hunting process in GWO. Equations 13, 14 and 15 are used to update the position of the best search agents.

$$\begin{aligned} D_\alpha &= |C_1.X_\alpha - X(t)| \\ D_\beta &= |C_2.X_\beta - X(t)| \\ D_\delta &= |C_3.X_\delta - X(t)| \end{aligned} \tag{13}$$

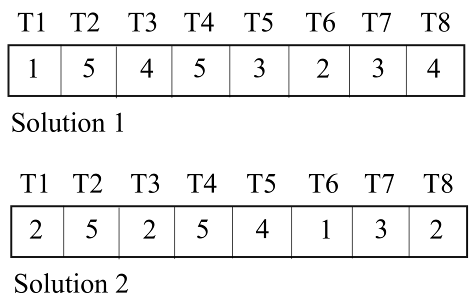
$$\begin{aligned} X_1 &= X_\alpha - A_1.D_\alpha \\ X_2 &= X_\beta - A_2.D_\beta \\ X_3 &= X_\delta - A_3.D_\delta \end{aligned} \tag{14}$$

$$X(t+1) = \frac{(X_1 + X_2 + X_3)}{3} \tag{15}$$

In equation 14, $X(t)$ is the position vector of grey wolf. X_1, X_2, and X_3 are position vectors of alpha, beta and delta wolves respectively.

3) Attacking Prey: The grey wolf attacks the prey until it stops moving. Mathematically, we decrease the value of a in each iteration. The controlling parameter a is defined in equation 16, where t is current iteration and N is the maximum number of iteration.

$$a = 2 \times (1 - \frac{t}{N}) \tag{16}$$



T1	T2	T3	T4	T5	T6	T7	T8
1	5	4	5	3	2	3	4

Solution 1

T1	T2	T3	T4	T5	T6	T7	T8
2	5	2	5	4	1	3	2

Solution 2

Fig. 3. Encoded Solution in Scheduling Problem.

IV. Proposed Algorithm

The complete description of the proposed algorithm is given in this section. The proposed algorithm, named HPSOGWO, is the combination of Particle Swarm Optimization and Grey Wolf Optimization. The basic idea of the HPSOGWO algorithm is to run the PSO algorithm for the first half of the total iterations and the best solution generated by the PSO ($gbest$) is initialized to the alpha wolf (α-wolf) and for the latter half of the total iterations run GWO algorithm. The best solution generated by the GWO is stored in alpha wolf and considered to be the best mapping of tasks and VMs. The complete algorithm is shown in Algorithm 1 and the major steps are shown in Fig. 5. The initial or range of the various parameters along with the explanation used in the proposed algorithm is shown in Table II.

A. Encoding the Scheduling Problem

The first step in applying the algorithm is to model the workflow scheduling problem. As discussed earlier, the scheduling problem is considered as a mapping between the user's tasks and virtual machines. The solution (particle and wolf) in the proposed algorithm can be represented using an array (or list). The array's index represents the tasks, and the value in the array represents the assigned VM. The similar encoding is used in [36] and it helps in the reduction of the complexity of the algorithm.

An example of the encoding of the solution is shown in Fig. 3. The array index represents the eight tasks (T1 to T8), and the value at each index represents a virtual machine or instance id. For Solution 1, the tasks T1 is allocated to VM1, T6 is assigned to VM2, T5 and T7 are assigned to VM3, T3 and T8 are allocated to VM4, and T2 and T4 are allocated to VM5. Similar is the case with Solution 2. This encoding does not deal with the precedence constraint of the workflow. For example, the tasks T1, T3, and T8 are assigned to VM2 which does not mean that T1 is executed first. In other words, it does not depicts the precedence among the tasks.

B. Initialize the Population

The HPSOGWO algorithm has a specific number of iterations; in our case, 500. The set of the solutions (particles) is known as the population. In the first iteration, the population is initialized with a random solution. The solution is improved with each iteration of the algorithm. A random initialization of population is illustrated in Fig. 4.

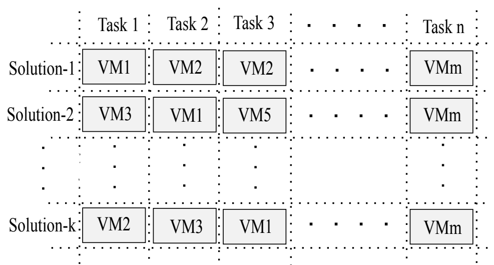


Fig. 4. Initializing the Population.

C. Evaluation of the Fitness Function

The algorithm begins with the calculation of execution time and assign these values into the execution time matrix as shown in equation 17. Each element value represent the execution time, for example $ET_{1,1}$ is the execution time of task T_1 on VM_1. The value of execution time in the matrix is calculated using equation 5.

$$ET-Mtx = \begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_n \end{array} \begin{array}{c} \begin{array}{cccc} VM_1 & VM_2 & \cdots & VM_m \end{array} \\ \begin{pmatrix} ET_{1,1} & ET_{1,2} & \dots & ET_{1,m} \\ ET_{2,1} & ET_{2,2} & \dots & ET_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ ET_{n,1} & ET_{n,2} & \dots & ET_{n,m} \end{pmatrix} \end{array} \tag{17}$$

The dependency of the tasks in a workflow can be represented using task dependency matrix (TD-Mtx) as shown in equation 18. Each element in the matrix is either 1 or 0. Suppose the value of $d_{1,2}$ is 1, then task T_2 is executed after task T_1.

$$TD-Mtx = \begin{array}{c} \\ T_1 \\ T_2 \\ \vdots \\ T_n \end{array} \begin{array}{c} \begin{array}{cccc} T_1 & T_2 & \cdots & T_n \end{array} \\ \begin{pmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,n} \\ d_{2,1} & d_{2,2} & \dots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n,1} & d_{n,2} & \dots & d_{n,n} \end{pmatrix} \end{array} \tag{18}$$

The cost matrix stores the cost of execution for a unit time of each VM as depicted in equation 19. $C_1, C_2, ...C_m$ are the unit execution costs of virtual machines $VM_1, VM_2..., VM_m$ respectively.

$$Cost-Mtx = (C_1 \quad C_2 \quad \cdots \quad C_m) \tag{19}$$

According to these matrices, we evaluate total execution time and total execution cost and the fitness function of each solution as mentioned in Section III-A.

D. Applying PSO Algorithm

The PSO algorithm starts with random population, and run for $n/2$ iterations, where n is the maximum iterations. PSO keeps track of the personal best ($pbest$) position and global best ($gbest$) position of the particle in each iteration. The updated position of the particles is influenced by $gbest$ and $pbest$ particle in each iteration to reach to the global best solution ($gbest$). The complete process is mentioned in Section III-B.

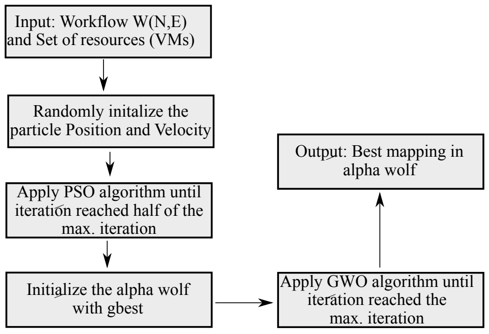

Input: Workflow W(N,E) and Set of resources (VMs)
Randomly initalize the particle Position and Velocity
Apply PSO algorithm until iteration reached half of the max. iteration
Initialize the alpha wolf with gbest
Apply GWO algorithm until iteration reached the max. iteration
Output: Best mapping in alpha wolf

Fig. 5. Major Steps of the Proposed Algorithm.

E. Applying GWO Algorithm

The best solution (α-wolf) of GWO algorithm is initialized with the best solution ($gbest$) obtained from the PSO algorithm. Then, we apply GWO algorithm for latter half of the iterations ($n/2+1$ to n) as mentioned in Section III-C. The α-wolf leads all other wolves to the better solution in every iteration of GWO algorithm. After meeting the stopping criteria, the optimal solution is present in α-wolf. And the tasks are assigned to the respective VM as suggested by the α-wolf.

Algorithm 1 The Proposed Algorithm.

```
1: Input: Workflows $W(N, E)$ and set of resources ($VM_1$, $VM_2$, ... $VM_j$)
2: Output: Best solution in alpha wolf. (mapping of tasks in $W$ with set of resources.)
3: Set the number of particles equal to the total number of tasks.
4: Randomly initialize the position and velocity of each particle.
5: Calculate the fitness value of each particle according to equation 1.
6: If the fitness value of the current particle is better than $pbest$ particle, set the current particle as new $pbest$.
7: After Steps 5 and 6 for all the particles, select the global best solution as $gbest$, among the $pbest$ particle.
8: For all particles, calculate velocity and update position as per equations 7 and 8.
9: Repeat from the Step 5 until the iteration is reached to the half of maximum iterations.
10: Set the number of wolves equal to the size of tasks.
11: Initialize alpha wolf position as $gbest$ and randomly initialize the other wolves position.
12: Initialize $A$, $C$, and $a$ as per equations 11, 12, and 16, respectively.
13: Calculate the fitness value of each wolf according to equation 1.
14: Calculate alpha wolf, beta wolf and delta wolf according to fitness value.
15: For all wolves, update position as per equations 15, 14, and 9.
16: Update $a$, $A$ and $C$.
17: Calculate the fitness value of all wolves.
18: Repeat from the Step 14, until maximum iterations are reached.
```

TABLE I. Simulation Parameters

Parameters	Values
Number of tasks	25 to 1000
Number of VMs	5
MIPS	1000
RAM	512 MB
Bandwidth	1000
Number of Processor	1
VM Policy	Time Shared

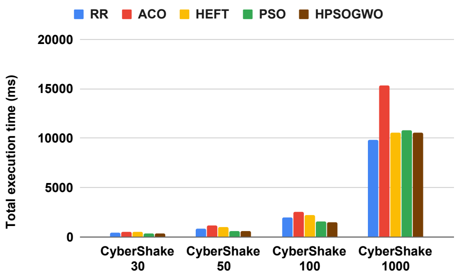


Fig. 6. Total Execution Time of Algorithms under CyberShake Workflow.

V. Performance Evaluation

A. Experimental Setup

The proposed algorithm (HPSOGWO) is executed under four scenarios to evaluate the total execution time and cost of the fitness function. The HPSOGWO algorithm is compared with the round-robin (RR) [37], ant colony optimization (ACO) [38], heterogeneous earliest time first (HEFT) [39], and particle swarm optimization (PSO) [12] algorithms. The four scenarios includes CyberShake, Montage, Inspiral, and Spith scientific workflows. These workflows are available with different tasks; for example, CyberShake is available with 30, 50, 100, and 1000 tasks. All experiment are carried out on a computer with Intel(R) Core i5-5200U CPU at 2.2.GHz, 4.00 GB of RAM, Windows 8 Pro 64-bit operating system. To simulate and evaluate the proposed algorithm's performance, we used the WorkflowSim-1.1 toolkit [40], which is an extension of CloudSim. Table I shows the simulation parameters used during the evaluation of the algorithm and Table II shows the initial values or the range of the values of different parameters used in the proposed algorithm.

B. Simulation Results

In this section, we do the performance comparison of the proposed algorithm, HPSOGWO, with the RR, ACO, HEFT, and PSO algorithms. The performance is measured in terms of total execution time (TET) and total execution cost (TEC) with the increasing number of tasks in the ranges from 25 to 1000 under four well known workflows: CyberShake, Inspiral, Montage, and Sipht. The maximum number of iterations were set to 500. Each scenario is executed 10 times and the average value of the result is considered. The simulation results are tabulated in Tables III, IV, and V.

1) Performance evaluation under CyberShake workflow: Fig. 6 shows the simulation results of the RR, ACO, HEFT,

TABLE II. PROPOSED ALGORITHM PARAMETERS.

Parameters	Values/Range	Explanation
Number of iterations	500	The total number of runs of the algorithm.
For PSO algorithm		
Particle Size	25	The number of particles, each particle represent a solution.
C_1,C_2	2	They are the acceleration coefficients.
w	0.1	It is inertia weight.
For GWO algorithm		
Wolf Size	25	The number of wolves, each wolf represent a solution.
a	[2,0]	It is descended from 2 to 0 during the algorithm iteration.
A	[-a,a]	Initially set to 0. It is used to model the convergence. If $A > 1$ wolves diverge from prey if $A < 1$ wolves converge to prey.
C	[0,2]	Initially set to 0. It is used to avoid falling into local optima.
D	Any value	It is mathematical model of surrounding the prey.
$r1$,$r2$	[0,1]	They are coefficients. The values are random in the range from 0 to 1.

TABLE III. TOTAL EXECUTION TIME OF DIFFERENT SCENARIOS.

Scenario	RR	ACO	HEFT	PSO	HPSOGWO
CyberShake 30	441.72	545.845	519.05	344.457	336.62
CyberShake 50	855.95	1150.6397	981.85	604.514	640.29
CyberShake 100	1941.49	2570.846	2205.28	1547.124	1480.97
CyberShake 1000	9842.98	15325.3909	10547.13	10765.049	10544.57
Inspiral 30	2177.64	2110.17	3981.31	2119.124	1930.611
Inspiral 50	3319.31	3718.62	3704.54	3123.322	2898.33
Inspiral 100	4779.51	7327.449	6196.76	4957.428	4725.432
Inspiral 1000	49640.257	60878.5568	54398.38	47443.998	47451.18
Sipht 30	4413.64	5206.29	5534.87	4431.95	4454.72
Sipht 60	8072.78	8942.873	10344.51	5943.129	5234.02
Sipht 100	10184.85	10518.799	15417.59	7653.846	6502.37
Sipht 1000	111822.22	119781.024	130992.15	89440.127	92791.18
Montage 25	57.13	66.266	56.88	67.854	61.861
Montage 50	132.23	139.977	130.35	135.604	137.631
Montage 100	259.83	276.801	257.72	273.306	267.8
Montage 1000	2559.28	2637.382	2559.35	2611.48	2635.56

TABLE IV. TOTAL EXECUTION COST OF DIFFERENT SCENARIOS.

Scenario	RR	ACO	HEFT	PSO	HPSOGWO
CyberShake 30	1803.04	1497.941	1817.27	917.385	897.406
CyberShake 50	3909.69	5053.1127	4180.39	2350.999	2218.277
CyberShake 100	9172.47	11691.154	10498.68	6527.428	6461.2
CyberShake 1000	48110.56	74273.85	51642.27	52036.77	50878.41
Inspiral 30	9007.73	8984.759	15072.03	7029.795	7035.689
Inspiral 50	14638.34	15092.29	15302.33	12525.773	12541.193
Inspiral 100	22711.35	28375.691	24897.88	21497.265	21969.247
Inspiral 1000	240473.418	243231.5666	266391.93	234635.922	233197.111
Sipht 30	17704.2	19575.234	11077.42	8165.034	6727.67
Sipht 60	36174.69	34359.755	30899.12	18551.321	18483.523
Sipht 100	46959.56	41439.9	60947.59	30556.972	26477.83
Sipht 1000	554868.95	585886.796	545244.42	438761.6	455008.262
Montage 25	264.65	284.834	248.92	260.19	255.212
Montage 50	628.34	636.823	570.58	584.654	572.457
Montage 100	1248.27	1304.69	1143.3	1226.514	1192.534
Montage 1000	12454.6	12605.791	11971.62	12304.51	12252.285

TABLE V. FITNESS VALUE OF DIFFERENT SCENARIOS.

Scenario	RR	ACO	HEFT	PSO	HPSOGWO
CyberShake 30	1122.38	1497.941	1168.16	630.926	617.01
CyberShake 50	2382.82	3101.88	2581.12	1477.76	1429.282
CyberShake 100	5556.98	7131	6351.98	4037.272	3971.086
CyberShake 1000	28976.77	44799.6217	31094.7	31400.91	30711.49
Inspiral 30	5592.685	5547.46	9526.67	4574.458	4483.157
Inspiral 50	8978.825	9405.455	9503.44	7824.548	7719.768
Inspiral 100	13745.43	17851.57	15547.32	13227.347	13347.343
Inspiral 1000	145056.84	152055.06	160395.15	141039.96	140324.15
Sipht 30	11058.92	12390.765	8306.14	6298.491	5591.192
Sipht 60	22123.73	21651.315	20621.81	12247.225	11858.77
Sipht 100	28572.2	25979.348	38182.59	19105.409	16490.1
Sipht 1000	333345.59	352833.91	338118.29	264100.864	273899.72
Montage 25	160.89	175.55	152.9	164.03	158.54
Montage 50	380.285	388.4	350.47	360.132	355.044
Montage 100	754.05	790.75	700.51	749.91	730.169
Montage 1000	7506.94	7621.5865	7265.48	7457.99	7443.923

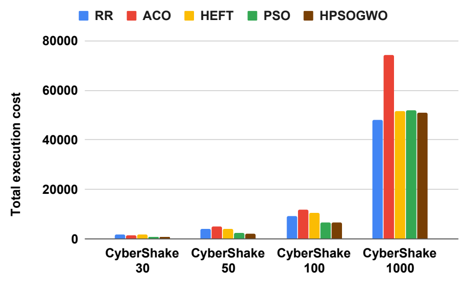


Fig. 7. Total Execution Cost of Algorithms under CyberShake Workflow.

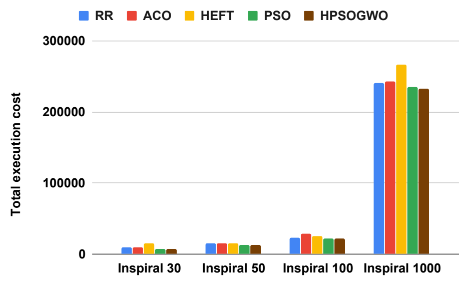


Fig. 9. Total Execution Cost of Algorithms under Inspiral Workflow.

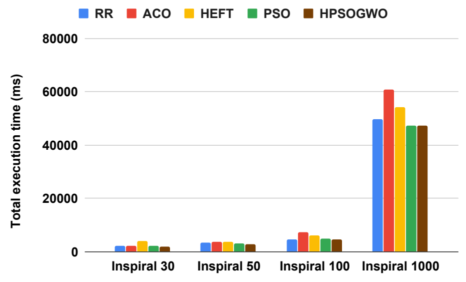


Fig. 8. Total Execution Time of Algorithms under Inspiral Workflow.

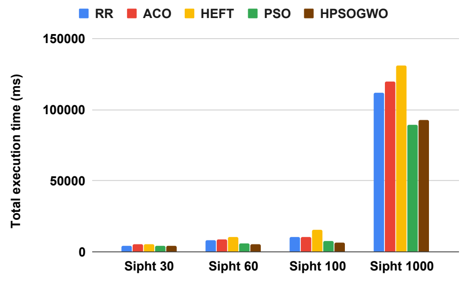


Fig. 10. Total Execution Time of Algorithms under Sipht Workflow.

PSO, and proposed algorithm (HPSOGWO) in term of total execution time (TET). The y-axis represents the total execution time, and the x-axis shows the number of tasks. For 30 tasks, the proposed algorithm decreases the TET by 2.33%, 54.19%, 62.16%, and 31.22% compared to PSO, HEFT, ACO, and RR, respectively. For 50 tasks, the proposed algorithm decreases the TET by 53.35%, 79.71%, and 33.68% compared to HEFT, ACO, and RR, respectively, but an increase of 5.59% in TET is noted compared to PSO. When the number of tasks are 100, the HPSOGWO reported 4.47%, 48.91%, 73.59%, 31.09% decrease in TET compared to PSO, HEFT, ACO, and RR, respectively. Similarly, for 1000 tasks, decrements of 2.09%, 0.02%, and 45.34% compared to PSO, HEFT, and ACO, respectively, but increase of 6.65% compared to RR was observed in TET.

Fig. 7 shows the total execution cost (TEC) of different algorithms under the CyberShake workflow with the increasing number of tasks. For 30 tasks, the HPSOGWO noted the decrease of 2.23%, 102.50%, 66.92%, and 100.92% in TEC compared to PSO, HEFT, ACO, and RR, respectively. While the TEC of HPSOGWO is decreased by 5.98%, 88.45%, 127.79%, and 76.25% compared to PSO, HEFT, ACO, and RR, respectively for 50 tasks. For 100 tasks, the TEC is decreased by 1.03%, 62.49%, 80.94%, and 41.96% compared to PSO, HEFT, ACO, and RR, respectively. The HPSOGWO algorithm lowers the TEC by 2.28%, 1.50%, 45.98%, and 5.44% compared to PSO, HEFT, ACO, and RR, respectively for 1000 tasks.

2) Performance evaluation under Inspiral workflow: Fig. 8 shows the performance of different algorithms in terms of total execution time with different number of tasks. The proposed algorithm performs well in almost all cases. The TET of the proposed algorithm is improved by 9.76%, 106.22%, 9.30%, and 12.7% compared to PSO, HEFT, ACO, and RR, respectively, for 30 tasks. Similarly, it is improved by 7.76%, 27.82%, 28.30%, and 14.52% compared to PSO, HEFT, ACO, and RR, respectively, for 50 tasks. When the number of tasks are 100, the TEC improvement of HPSOGWO is 4.91%, 31.14%, 55.06%, and 1.14% compared to PSO, HEFT, ACO, and RR, respectively. The proposed algorithm's TEC is decreased by 14.64%, 28.29%, and 4.61% compared to HEFT, ACO, and RR, respectively for 1000 tasks. However, a slight increase in TET of 0.01% is observed compared to PSO for 1000 tasks.

As depicted in Fig. 9, there is an improvement in the TEC by 114.22%, 27.70%, and 28.03% in the HPSOGWO compared to HEFT, ACO, and RR respectively, while the deterioration of 0.08% in performance is reported compared to PSO, for 30 tasks. When there are 50 tasks, the proposed algorithm's performance is improved by 22.02%, 20.34%, and 16.72% compared to HEFT, ACO, and RR, respectively, while deterioration of 0.12% in performance is observed compared to PSO. For 100 tasks, the TEC of HPSOGWO is declined by 13.33%, 29.16%, and 3.38% compared to HEFT, ACO, and RR, respectively. Also, the performance of HPSOGWO is decreased by 2.15% compared to PSO. For 1000 tasks, the HPSOGWO outperformed compared to other compared algorithms. There is 0.62%, 14.23%, 4.30%, and 3.12% of improvement in TEC compared to PSO, HEFT, ACO, and RR, respectively.

3) Performance evaluation under Sipht workflow: Fig. 10 shows the total execution time of different algorithms under sipht workflow with a different number of tasks. The performance of the proposed algorithm for 30 tasks is better than that of HEFT and ACO, but compared to PSO and RR, it decreases a little. For 60 and 100 tasks, HPSOGWO performed well compared to other algorithms. For 1000 tasks, the performance

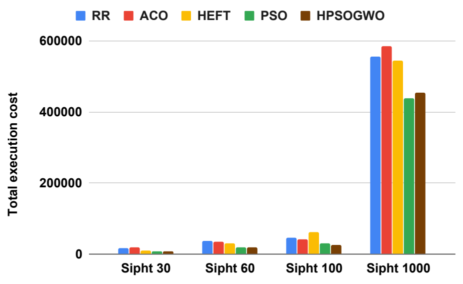


Fig. 11. Total Execution Cost of Algorithms under Sipht Workflow.

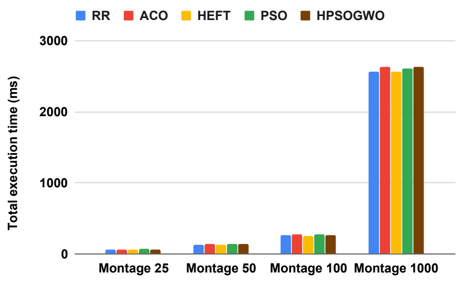


Fig. 12. Total Execution Time of Algorithms under Montage Workflow.

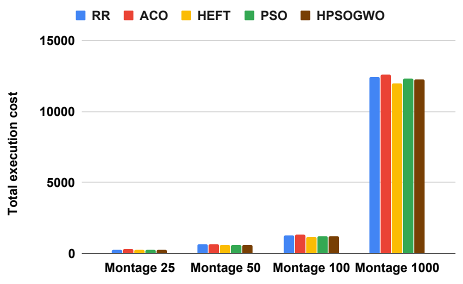


Fig. 13. Total Execution Cost of Algorithms under Montage Workflow.

is better than HEFT, ACO, and RR, but a slight increment of 3.61% in TET is observed compared to PSO.

The comparison among the total execution cost of different algorithms under sipht workflow is shown in Fig. 11. The proposed algorithm outperforms the other algorithms for all the cases with up to 190.97% of decrements in TEC. Except for 1000 tasks, an increment of 3.57% in TEC is observed in the proposed algorithm compared to the PSO algorithm.

4) Performance evaluation under Montage workflow: The performance in terms of total execution time and total execution cost with 25, 50, 100, and 1000 tasks under montage workflow are shown in Fig. 12 and Fig. 13 respectively. The proposed algorithm outperforms in all the cases compared to the ACO algorithm, with up to 7.12% reduction in TET. For 25 and 100 tasks, the declines of 9.69% and 2.06% are noticed in total execution time compared to PSO. While for other cases, the proposed algorithm did not perform well. The TEC is reduced for HPSOGWO compared to PSO, ACO, and RR. The HPSOGWO does not perform well compared to HEFT, with up to 4.13% of increment in TEC.

VI. CONCLUSION

A novel hybrid meta-heuristic algorithm based on a multi-objective model called HPSOGWO is proposed in the present paper. The proposed algorithm is the hybrid version of Particle Swarm Optimisation (PSO) and Grey Wolf Optimisation (GWO) algorithms. The objectives of the proposed algorithm are to optimize the total execution cost and total execution time. The HPSOGWO algorithm is tested on the four scientific workflows: Montage, CyberShake, Inspiral, and Sipht with different number of tasks. The experimental results shows that the proposed algorithm reduces the total execution time and cost compared to PSO, HEFT, ACO, and RR algorithms. In future work, some other parameters like total energy consumption, load balancing, response time, etc. will be considered for the evaluation purpose. The other algorithms can be considered for making the new hybrid algorithm and evaluate under the same parameters.

REFERENCES

[1] K. Dasgupta, B. Mandal, P. Dutta, and S. Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing," *Procedia Technology*, vol. 10, pp. 340–347, jan 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212017313005318

[2] A. Mazrekaj, I. Shabani, and B. Sejdiu, "Pricing Schemes in Cloud Computing: An Overview," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 80–86, 2016.

[3] G. Natesan and A. Chokkalingam, "Multi-Objective Task Scheduling Using Hybrid Whale Genetic Optimization Algorithm in Heterogeneous Computing Environment," *Wireless Personal Communications*, 2019.

[4] R. Ferreira da Silva, H. Casanova, A. C. Orgerie, R. Tanaka, E. Deelman, and F. Suter, "Characterizing, Modeling, and Accurately Simulating Power and Energy Consumption of I/O-intensive Scientific Workflows," *Journal of Computational Science*, vol. 44, 2020.

[5] G. Juve and E. Deelman, "Scientific Workflows in the Cloud," *Grids, Clouds and Virtualization*, pp. 71–91, 2011.

[6] E. Deelman, G. Singh, M. H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, 2005.

[7] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, 2013.

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

[9] S. Saeedi, R. Khorsand, S. Ghandi Bidgoli, and M. Ramezanpour, "Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing," *Computers and Industrial Engineering*, vol. 147, no. June, p. 106649, 2020. [Online]. Available: https://doi.org/10.1016/j.cie.2020.106649

[10] A. Kamalinia and A. Ghaffari, "Hybrid Task Scheduling Method for Cloud Computing by Genetic and DE Algorithms," *Wireless Personal Communications*, 2017.

[11] J. Yu and R. Buyya, "A budget constrained scheduling of workflow applications on utility grids using genetic algorithms," *2006 Workshop on Workflows in Support of Large-Scale Science, WORKS '06*, 2006.

[12] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, 2010.

[13] M. Hosseini Shirvani, "A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems," *Engineering Applications of Artificial Intelligence*, vol. 90, no. September 2019, p. 103501, 2020. [Online]. Available: https://doi.org/10.1016/j.engappai.2020.103501

[14] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” *Advances in Engineering Software*, 2014.

[15] A. Khalili and S. M. Babamir, “Optimal scheduling workflows in cloud computing environment using Pareto-based Grey Wolf Optimizer,” *Concurrency Computation*, vol. 29, no. 11, pp. 1–11, 2017.

[16] L. Zhang, L. Zhou, and A. Salah, “Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments,” *Information Sciences*, vol. 531, pp. 31–46, 2020. [Online]. Available: https://doi.org/10.1016/j.ins.2020.04.039

[17] K. Dubey, M. Kumar, and S. C. Sharma, “Modified HEFT Algorithm for Task Scheduling in Cloud Environment,” *Procedia Computer Science*, vol. 125, pp. 725–732, 2018.

[18] G. Patel, R. Mehta, and U. Bhoi, “Enhanced Load Balanced Min-min Algorithm for Static Meta Task Scheduling in Cloud Computing,” in *Procedia Computer Science*, 2015.

[19] R. Vijayalakshmi and V. Vasudevan, “Static batch mode heuristic algorithm for mapping independent tasks in computational grid,” 2015.

[20] M. M. Golchi, S. Saraeian, and M. Heydari, “A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation,” *Computer Networks*, vol. 162, p. 106860, 2019. [Online]. Available: https://doi.org/10.1016/j.comnet.2019.106860

[21] K. L. Eng, A. Muhammed, M. A. Mohamed, and S. Hasan, “A hybrid heuristic of Variable Neighbourhood Descent and Great Deluge algorithm for efficient task scheduling in Grid computing,” *European Journal of Operational Research*, 2019.

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

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

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

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

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

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

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

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

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

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

[32] S. Mirjalili, S. Saremi, and S. Mohammad, “Multi-objective grey wolf optimizer : A novel algorithm for multi-criterion optimization,” *Expert Systems With Applications*, vol. 47, pp. 106–119, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2015.10.039

[33] A. Pasdar, Y. C. Lee, and K. Almi’ani, “Hybrid scheduling for scientific workflows on hybrid clouds,” *Computer Networks*, vol. 181, no. August, p. 107438, 2020. [Online]. Available: https://doi.org/10.1016/j.comnet.2020.107438

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

[35] J. Kennedy and R. Eberhart, “Particle Swarm Optimization, Proceedings of IEEE International Conference on Neural Networks Vol. IV: 1942–1948.” 1995.

[36] T. P. Pham and T. Fahringer, “Evolutionary Multi-objective Workflow Scheduling for Volatile Resources in the Cloud,” *IEEE Transactions on Cloud Computing*, vol. 7161, no. c, pp. 1–12, 2020.

[37] T. Ghafarian, B. Javadi, and R. Buyya, “Decentralised workflow scheduling in volunteer computing systems,” *International Journal of Parallel, Emergent and Distributed Systems*, vol. 30, no. 5, pp. 343–365, 2015.

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

[39] H. Topcuoglu, S. Hariri, and M. Y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[40] W. Chen and E. Deelman, “WorkflowSim: A toolkit for simulating scientific workflows in distributed environments,” in *2012 IEEE 8th International Conference on E-Science, e-Science 2012*, 2012.