A Hybrid-based Multi-criteria Approach, for Efficient Workflow Tasks Assignment in Cloud Computing Environment

J. Kok Konjaang
Department of Computing and Information Technology
Bolgatanga Technical University, Ghana

Emmanuel Bugingo
College of Business and Economics
Department of Business Information Technology
University of Rwanda

David Sanka Laar
Department of Information Systems and Technology
University of Technology and Applied Sciences, Navrongo, Ghana

Elisha A. Ayakpagi
Information & Communication Technology Directorate
Bolgatanga Technical University, Ghana

ABSTRACT
Cloud computing has emerged as a distribution hub, that offers ubiquitous access, to a shared pool of cloud resources for modeling, executing, and performing big data analyses of businesses as well as scientific workflow applications. The workflow scheduling process is to allocate resources for users' tasks in a way that satisfies the constraints while optimizing some objectives. One of the most challenging problems in workflow scheduling is determining an optimal number of VMs configurations that can schedule streaming tasks to speed up the execution time while incurring a relatively low cost and energy consumption. Several methodologies have been proposed with the aim of generating good schedules to improve workflow execution. However, most of these existing methodologies on workflow scheduling are improving on relatively simple objectives (cost and makespan) without looking at the bigger problem (VM provisioning), which is mostly the prime causes of high execution cost, makespan, and energy consumption. In this paper, we proposed a novel hybrid-based multi-criteria deadline-constrained scheduler, known as MOW-PSO. MOW-PSO is a fusion of PSO and MOWOS, aiming to determine an optimal number of VMs configuration for a proper task to VM mapping, that finds a suitable solution to the three important yet, conflicting scheduling objectives; energy consumption, execution makespan, and cost without violating user-defined deadlines and budget constraints. Our approach is evaluated using five different representative workflows with four different workload patterns through WorkflowSim. The results prove the effectiveness of our proposed approach over the state-of-the-art algorithms.

General Terms
Workflow Scheduling

Keywords
Cloud Computing, Heuristics, Meta-heuristics, Hybrid, VM Provisioning, Workflows Scheduling.

1. INTRODUCTION
Highly multi-tenant and scalable platforms such as cloud computing attract a lot of researchers from the relevant research community [1, 2, 3, 4]. One reason for this attraction is to provide a good resource provisioning plan which has been a major challenge in cloud data centers for several years. This problem has caused a lot of setbacks in workflow scheduling, including tasks missing their deadlines [5, 6], high execution cost [7], over-consumption of energy [8], etc. In the cloud data center, there are only two parties of interest, that is, the service provider and the service user. The service provider is interested in making a profit by ensuring efficient utilization of all resources to increase throughput [9] while the user's interest is to optimize their application performance to ensure value for money [10]. However, a lack of a proper plan for resource provisioning can result in dissatisfaction between these two parties of different interests. The resource provisioning problem can be traced back to [11, 12, 13], among which the disparity in VM configuration and distribution is the major cause for over-consumption of energy in cloud data centers. This has been confirmed in [14, 15, 16, 17], that a poor resource provisioning plan causes under-utilization of cloud resources, unbalanced workloads, and high energy consumption. Moreover, it has been reported that a lack of intelligent VM provisioning strategies will mean all the VMs configured on a single CPU will not get the needed processing power to execute the tasks to reduce the scheduling length, which is a violation in the agreed SLAs [18]. This could lead to a reduction in system performance or cause a complete failure during workflow execution in a worst-case scenario.

Cloud computing has emerged as a multi-tenant and scalable application infrastructure for processing large-scale applications such as workflows. Workflows are used to analyze Big Data problems in a complex scientific computing system [19]. They provide a structured and distributed ordering through which scientific problems are solved in various scientific domains such as biology, physics, astronomy, and health sciences [20, 21, 22, 23, 24]. They are modeled as directed acycle graphs (DAGs) with hundreds or thousands of interdependent tasks and deployed on an Infrastructure as a Service (IaaS) cloud [25] by efficient and robust scheduling algorithms for scheduling.

Workflow scheduling has become an important research area defined by many as an NP-complete problem [26, 27, 28]. Many algorithms have been proposed to find an optimal solution to the workflow scheduling problem. Some of these researchers have applied meta-heuristic algorithms, such as [29, 30, 31], other researchers such as [25, 32, 33] have applied heuristics, while others have introduced hybrid algorithms, such as the works in [34, 35, 36]. However, resource provisioning, which is a major challenge in scheduling workflow tasks, is not fully considered by these researchers. Cloud service providers deploy VMs with various configuration settings that help them compute the services they offer. One of these configuration settings is the CPU processing power. When selecting an optimal VM configuration, it is important to examine the capacity of the VM relative to the size of the application. Depending on the size of the application, it is important to have a certain number of VMs to speed up the execution time of tasks through parallelism. The provided sum of CPU processing power must be divided among a certain number of VMs. However, there is a big problem in choosing the number of VMs. A small number of VMs can speed up the execution time of the workflow application. This is because each VM in the group gets a larger share of the CPU processing power to use. However, this process can result in a lot of idle time due to the dependencies between tasks. A large number of VMs can help execute many tasks simultaneously, but this increases execution cost and energy consumption and slows down execution makespan. This is because each VM gets only a small fraction of the CPU processing power to execute the running tasks. Thus, finding an optimal number of VMs to generate a schedule in a way that satisfies user constraints while conserving energy, balancing workload on resources and reducing execution cost and makespan is challenging.

In this research, we consider the VM provisioning as a major challenge in scheduling heterogeneous workflows on heterogeneous cloud resources to improve cloud users QoS. To this end, we proposed a hybrid-based multi-criteria particle swarm optimization method (MOW-PSO) based on the capabilities of our previous method, the Multi-Objective Workflow Optimization Strategy (MOWOS) [37]. In MOWOS, we have shown that tasks splitting and proper VM selecting can improve workflow scheduling. MOW-PSO is a fusion of PSO and MOWOS, which has a similar architecture to the PSO algorithm. It is designed to find an optimal number of VMs that can be used to generate a schedule that maximizes energy conservation as well as reduces execution cost and makespan. Recently, several hybrid studies have been conducted that aim to improve workflow scheduling. For example, Choudhary, et al. [34] presented a hybrid-based approach called HGSA, for scheduling workflow tasks to reduce monetary cost ratio and schedule length ratio. Similarly, hybrid chemical reaction optimization (HCRO) was introduced by Xu, Yuming, et al. [38]. HCRO is efficient in reducing the schedule length (makespan) of workflow tasks; however, our approach is different from these approaches. In our approach, we consider the processing power of the CPU configuration settings, which helps in determining the number of VMs that can be configured on a CPU to speed up the execution time of the workflow application. Other factors considered in the proposed method are the size of the workflow application to be scheduled, the types of VMs and their execution capacities, and finally optimizing multi-objectives. However, previous works have not considered most of these factors, which are important for improving workflow scheduling. Our approach improves the particle search efficiency and convergence rate. The main contributions of MOW-PSO include:

1. A scalable algorithm that combines heuristics and meta-heuristics, capable of addressing resource pro65 visioning problems in the cloud data center.

2. A novel scheduling model that leverages Particle Swarm Optimization to generate a schedule at a reduced cost and time.

3. A novel measure that distributes workloads on cloud resources evenly for effective load balancing, and facilitating resource provisioning decisions.

The remainder of the article is organized as follows: Section 2 contains a description of related work on workflow scheduling in cloud computing. Section 3 contains the mathematical formulation of the proposed method, the system model, the energy model and the application model. The proposed MOW-PSO algorithm is presented in Section 4. Section 5 presents the performance evaluation and the analysis of the results. Finally, in Section 6, the conclusions and future work are presented.

2. RELATED WORK

In workflow scheduling, the real-world problems of workflow tasks execution are faced with multi-objective circumstances in which many of these objectives conflict with each other [39]. As a result, developing scheduling schemes with a good resource provisioning plan, that selects an optimal number of VMs to schedule a workflow task from the heterogeneous workflows, regardless of the type and workload, such that 80 tasks to resources mapping produce a schedule that reduces energy consumption, makespan, and execution cost, without violating user-defined deadlines and budgets is challenging.

Many works have been proposed, from static resource provisioning techniques [40, 41, 42] to dynamic resource provisioning [43, 5, 44, 14] techniques to improve tasks scheduling in the cloud computing environment. Some of these works presented heuristic based-methods [45, 46, 47], while others presented meta-heuristic based methods [48, 49, 50, 51, 52]. Heuristic algorithms are developed to find an optimal or near-optimal solution to a task scheduling problem without guaranteeing that it is optimal or rational, but sufficient to achieve the set goal [53]. They are designed to solve problematic issues more accurately and faster than meta-heuristic algorithms [54]. In general, heuristic algorithms depend on the problem at hand and rely on past knowledge and exploration to find an optimal solution to a problem. However, they are inherently 90 greedy and may be trapped in a local optimum [55]. An advantage of heuristic-based scheduling methods over meta-heuristic-based methods is that they are efficient and can provide good solutions [56]. Metaheuristic-based methods, on the other hand, are nature-inspired algorithms [57, 58] designed for complex optimization problems [59]. Meta-heuristic algorithms are capable of finding solutions directly in the search space; however, they are time-consuming [60].

Meta-heuristic algorithms have emerged as the best method for workflow scheduling [35, 61, 14, 62, 63]. Some recent works on meta-heuristic includes: Shishido, et al. [49] who investigated on the effectiveness of meta-heuristic algorithms in generating effective schedules in the cloud computing environment. The study focused on evaluating the impact of the two most commonly used meta-heuristics, i.e., PSO and GA. They concluded that meta-heuristic methods are efficient in generating better schedules that minimize workflow execution

cost and response time. However, meta-heuristic methods consume more time [27]. A concept of Harmony Search (HS) algorithm known as polyrhythmic Harmony Search (PHS) for scientific workflow scheduling was proposed in Melnik and Trofimenko [50] to enable the use of two heterogeneous harmonies, namely scheduling tasks and optimizing tasks in the computing environment to shorting tasks execution length. The workflow scheduling problem is improved in Wu et al [51] where a Revised Discrete 105 Particle Swarm Optimization (RDPSO) was proposed. The proposed work considers two important workflow scheduling parameters, namely, transmission cost, and communication cost, with the aim of minimizing the makespan and cost of scheduling workflow tasks. Based on experiment, RDPSO is able to save cost and achieve better performance of makespan than PSO and BRS. However, the RDPSO algorithm is not efficient for a large search space [64].

Recently, some researchers identified time complexity as a drawback of meta-heuristics. In their view, heuristic-based approaches are preferable to the workflow scheduling problem [65, 58, 66]. For example, the literature in [45] uses a heuristic algorithm in heterogeneous computing environments to minimize the execution cost, execution makespan, and energy consumption of a running workflow application. In [67], a Partition Problem-based Dynamic Provisioning and Scheduling (PPDPS) approach that minimizes workflow execution cost with deadline constraints was used. Ndamlabin et al [75] used Cost-Time Trade-off efficient Workflow Scheduling (CTTWS) algorithm to improve the workflow scheduling with deadline constraints. However, the scheduling models for the above methods are different from ours. In our algorithm, we combine the advantages of both meta-heuristic and heuristic to generate effective schedules. This makes ours more robust in dealing with different task scheduling issues.

The two most popular based and most frequently used heuristics, Heterogeneous Earliest Finish Time (HEFT) and Critical-Path-on-a-Processor (CPOP) algorithms were proposed in [68] to improve CPU efficiency in a heterogeneous multiprocessor system. The HEFT algorithm is used to minimize the finishing time of tasks by selecting and assigning tasks with the highest upward rank values to a processor, whiles the CPOP algorithm assigns tasks using the summation of upward and downward values for priorities. Nevertheless, their approach did not consider the workload of the CPU and how it can be configured to improve scheduling efficiency. Another attempt to improve workflow scheduling is proposed in Haidri, et al [46], where tasks with maximum upward priorities are mapped on the lower cost virtual machine. Results in [46] were improved for overall execution time and economic cost while meeting deadline and priority constraints. However, the proposed method did not consider the job completion delays and its impact on resources.

Apart from heuristic and meta-heuristic algorithms, other researchers in the workflow scheduling community are investigating the workflow scheduling issue and proposing methods that combine two algorithms in solving the workflow scheduling problem. When two or more methods are combined, they are referred to as a hybrid or hybridized approach. The hybrid methods are believed to be more robust in handling the workflow scheduling problem [35]. Among the algorithms proposing hybrid methods includes [64] that proposed a hybrid GA-PSO algorithm to effectively schedule workflow tasks to reduce tasks execution cost, makespan, and ensure all workloads are well distributed. Also, SJF and RR with dynamic quantum hybrid algorithm (SRDQ) [69] is a task scheduling algorithm that combines Shortest-Job-First (SJF) and Round Robin (RR) algorithms to schedule tasks in the cloud environment. SRDQ can reduce waiting time and response time through the use of dynamic task quantum to balance the waiting time interval between the long and the shot tasks. However, SRDQ did not consider energy consumption as a parameter. Furthermore, Mirsaeid Hosseini [70] proposed a hybrid discrete particle swarm optimization (HDPSO) algorithm for workflow execution using novel theorems in the first phase to produce swarm members as input. In the second phase, DPSO is applied to produce a balance between exploration and exploitation, and finally, in the third phase, Hill Climbing technique is applied to improve the overall scheduling performance. Finally, in [34], a hybrid gravitational search algorithm (HGSA) was used for scheduling. The technique deployed cost/time equivalence procedure to eliminate every inferior agent, thereby minimizing tasks execution makespan and cost, however, the time complexity of HGSA is high.

Analyzing the previous works as presented in this section shows that many approaches have been carried out, from heuristics [45, 68, 46], meta-heuristics [49, 27, 50], to hybrid algorithms [64, 71, 69] to improve workflow scheduling, however, none of these methods is promising a guaranteed solution to the workflow scheduling problem. This, therefore, necessitated this research to explore other options to improve workflow scheduling. In this paper, our goal is to propose a novel hybrid low-time complexity scheduling algorithm that considers CPU processing power as a benchmark for determining the best optimal VM configuration (number and type) on each CPU to speed up the execution time of tasks through parallelism.

3. SYSTEM MODEL AND PROBLEM DEFINITION

In this section, we present the workflow model, cloud resource model, task to VM assignment model, and formulation of the multi-criteria problem.

3.1. Workflow model:

Workflow is modeled as a DAG, which is defined as W= (T,E). Where E is a set of "n" tasks representing tasks dependency. Task dependency is defined as =(t_i , t_j), where $t_i \in T$ is a predecessor task of $tj \in T$ and $ti \neq tj$ is called the immediate successors of workflow task tj , thus, workflow task tj which is a child task cannot be executed until workflow ti , a parent task is completely executed. Due to the structuring and ordering that exits in workflow execution, a task is only allowed to be executed when all its parent's tasks have been successfully executed and it's required data reported. However, if there exists any task in the flow with no parent, call that task an entry task, and call any task without a child an exit task. A sample workflow DAG of 12 workflow tasks with entry and exit tasks is depicted in Figure 1. We can therefore conclude that there are only four different types of tasks in every workflow which are simplified below:

- Predecessor task = pre_{task}
- Successor task = $succ_{task}$
- Entry task = $entry_{task}$
- Exit task = $exit_{task}$

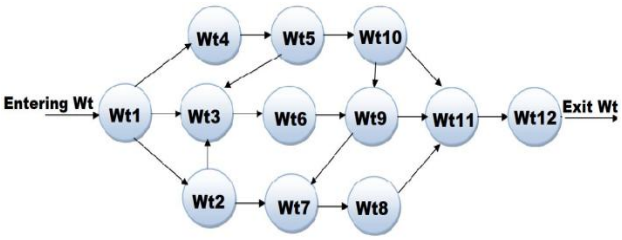

Figure 1: Sample Workflow Dag with Entering and Exit Workflow Tasks

3.2. Cloud Resource Model

The cloud resource model in Figure 2 is a modified model derived from our previous work [37]. The model consists of several cloud users and different cloud service providers that offer cloud resources (VMs). Let r = $= \bigcup_{s=1}^{\infty} \{r_s\}$ represent the VMs in the cloud data center which is unlimited. Let $K = \bigcup_{k=1}^{n} \{R_k\}$ denote the types of VMs, where n is the number of VMs in type k [72] which is represented as R = (vm1, vm2, vm3 .....vmk) be the list of cloud resources for workflow tasks scheduling. These instances are known to be running VMs that come with different types in terms of computing capacity, bandwidth, memory, etc. These instances will have to be configured on CPUs provided by the cloud service providers. The capacity of the CPU Vis-`a-vis the number of VMs that will be configured on a CPU will determine the execution time of the VM. A small number of VMs on a CPU can speed up the execution time of the workflow application. This is because each VM in the group is given a larger share of the CPU processing power. This process can result in a lot of idle time due to the dependencies between tasks. On the other hand, a large number of VMs can help execute many tasks simultaneously through parallelization, but this increases execution cost and energy 185 consumption and slows down the execution makespan. The execution process is slowed because; every VM receives only a small fraction of the CPU processing power to execute the running tasks. Due to the challenge of selecting an optimal VM configuration, our model is proposed in the following steps: (1) determine the number of tasks and their execution length, (2) determine the number of VM and their execution capacity, (3) determine the total CPU capacity required, and (4) determine the number of VM configuration based on the tasks and their user-defined deadline. Depending on the size of a task, a determination can be made on the number of VMs to be deployed on a CPU to speed up the execution time of workflow tasks, while incurring a relatively low cost.

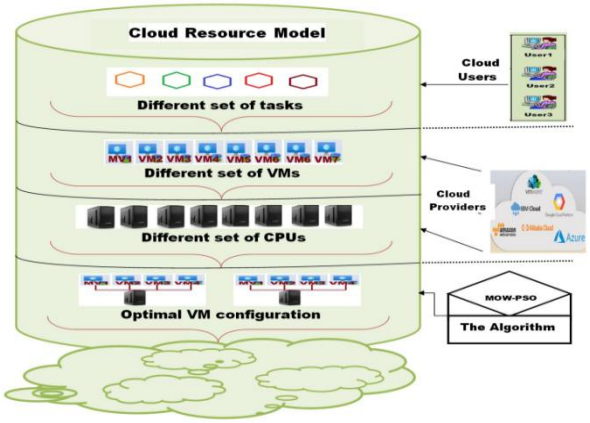

Fig 2: Cloud Resource Model

3.3. Task to VM Assignment Model

Scheduling tasks on cloud computing requires a set of different cloud resources with different types, speeds, and costs to map different workflow tasks from different scientific domains (bioinformatics, astrophysics, earthquake) to provide quality solutions. However, the concern has always been how to develop a low-time complexity scheduler with a well-defined plan to determine the most optimal VM configurations. This paper developed a task to VM assignment model depicted in Figure 3. The model comprises the major actors in the workflow scheduling environment, including the service users who present their request for scheduling, and the cloud service provider (resource owner) who provides the resources for scheduling to take place. However, before scheduling could commence, both parties must agree on the type of service, the quality of service, etc. to be provided in the service negotiation process called the service level agreement (SLA). Once negotiations are concluded, the user(s) request (task) is sent. The request will go through the scheduling process until the scheduler finds a suitable resource (VM) for the task(s) to be executed. This process is continued until all the tasks are scheduled and a report (feedback) is sent back to the user.

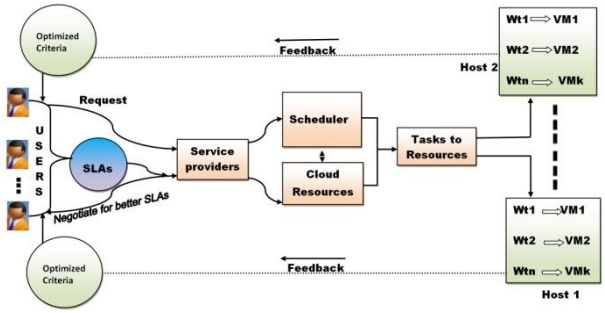

Fig 3: Task to VM assignment Model

3.4. Energy Conserving Model

Energy consumption has become a challenge for industrialization. This is as a result of the ever-growing demand for high processing system for data processing. Our energy consumption model is presented in Figure 4. It is modeled based on the system capability, the type of request (large or small), number of deployed VMs and their power consumption level. Normally, is it stated that, the higher number of processors used in a data center, the higher the energy consumed at the data center [73] .This is because each processor deployed in a data center will need energy to function, therefore, the more processors deployed, greater energy is consumed. Example, they are n number of cloud service providers (CSP) offering different cloud resources (VMs) which is represented (csp1, csp2, csp3 ..... cspn). Let r = represent a set of cloud resources with different processing capacities which is unlimited in a data center for users. Let k = be the type of VMs, where n is the number of VMs deployed to map users request. The power consumed by these VMs will depend on the type of processors and the number of processors used which is represented by p(n).

Total power consumption in data centre includes Dynamic deployment and switching of VM (DP), Idle VM (IVM), number of Processors deployed (PD) and Work Load (WL) which is modelled mathematically in question 1.

TPC = DP + IVM + PD/WL (1)

Definition 1: Dynamic deployment and switching of VM (DP): This is where all the VM are dynamically provisioned, switch off/on and configured for use without the input of the user. Significant power can be conserved if algorithms can be

designed to limit needless deployment of VMs and avoid unnecessary switching activities.

Definition 2: Idle VM (IVM), A VM is said to be busy when it is currently executing or processing a workflow task. Idle VMs, on the other hand, are VMs that have not yet been assigned to any job or have completed the execution process and are waiting for more jobs to be executed. The model checks, if there are some idle VMs, and if so, it switches all idle VMs to the memory state. To do this, Kernel-based processing node was used to build a virtual management policy. This policy assigns time frame to each VM, thus, every VM is configured to automatically switch to power saving mode, if the processing node is idle for 5 minutes. Therefore, to obtain an optimal energy consumed in data center, we first adopted the total power consumption as in equation 2. as presented by [73]. where Pkidle and Pkmax are the power consumed by the processor when idle, and at 100 % utilization respectively, whereas uk(t) is the utilization rate of the processor, which is a function of the time. Therefore, to obtain optimal energy consumption in data center, let pnu be the number of used processors, pnidle be the number of idle processors. Optimal Power consumption is modeled in equation 2.

$$Optp = pn + (pnud - pnidle) \times js. \quad (2)$$

Where optp is the optimal power, pn is number of processor deployed, pn_{up} is the number of used processors, pn_{idle} is the number of idle processors and js the job size.

3.5. Problem formulation

We first look at the workflow application which is modeled as a DAG, which involves workflow tasks with dependency among them. The workflow tasks are required to be executed on working VMs with different execution capacities. In a cloud data centre, there are two parties of interest, the service provider and the service user. The service provider is interested in optimizing resources utilization, while the user is interested in optimizing the performance of their application to ensure value for money. However, the lack of a proper 245 resource provisioning plan can lead to dissatisfaction between these two parties with different interests. For example, given a workflow, a set of VMs, and a sum of CPU processing power that is fixed for the entire execution period, providers may want to manage their resources by configuring many VMs on a single CPU. However, placing many VMs on a single CPU slows down the execution process because each VM in this group receives only a small portion of the CPU's processing power to execute the running tasks. This can lead to a reduction in system performance or, in the worst case, a complete failure during workflow execution, resulting in many tasks missing their deadlines, which is a violation of the agreed service level agreement (SLA). The problem is how algorithms can be designed to determine optimal VM configuration that will schedule tasks to produce good quality results. The goal of this paper is to develop a hybrid workflow scheduler with a good resource provisioning plan that can produce a suitable schedule to minimize workflow execution cost, energy consumption as well as execution makespan. User-defined deadlines (δ) and budget (B) are the main constraints [14].

The problem is formulated as follows: How to develop a hybrid scheduling algorithm with a good resource provisioning plan to determine optimal number of VMs to be configured on CPU to produce good schedules that minimize tasks execution cost, energy consumption and makespan under the user-defined budget and 260 deadline?

The problem is formulated mathematically as in equation 3.

$$optNbVMs = \begin{cases} AvgW, if\, AvgW \le StdDW \\ min\{AvgW + StdDW, MaxW\}, otherwise \\ Minimize \begin{cases} T_{Makespan} \\ TE_{Cost} \\ T_{Energy} \end{cases} \end{cases} \quad 3$$

where opNbVMs is the optimum number of VMs, MaxW is the maximum, AvgW is the average and StdDW is the standard deviation of the levels' width of the workflow. $T_{Makespan}$, TE_{Cost}, T_{Energy} are the Total makespan, Total execution cost and Total energy to be minimized respectfully.

4. THE PROPOSED HYBRID MOW-PSO ALGORITHM

We began with a model depicted in Figure 4, which defines the working schemes of the proposed MOWPSO algorithm. The proposed algorithm is supported by three (3) other algorithms, as stated in the model. These include optimal number of VMs determination algorithm (ON-VM-DA), MOWOS algorithm, and finally the usage of PSO algorithm to converge onto an optimal solution.

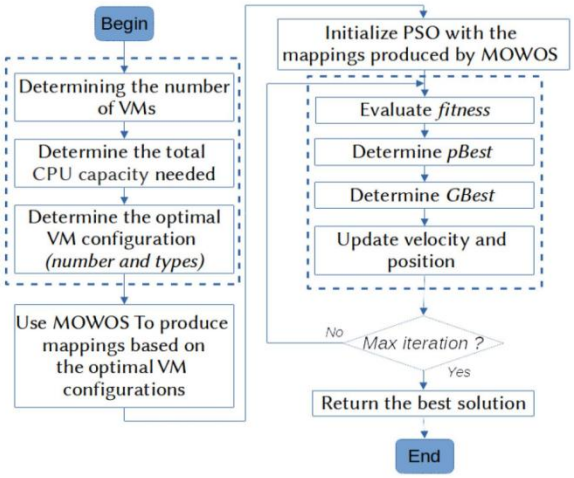

Fig 5: MOW-PSO Model

4.1 ON-VM-DA Algorithm

This algorithm solved the optimum number of VM configuration problems in three steps. First, we determined the number of VMs to use. Secondly, we determined the total CPU frequency. Thirdly, we determine the optimal VM configuration (number and types of VMs). For that, the type of each VM needs to be known.

4.1.1. Determining the number of VMs

Determining the number of VMs to use for the execution of a workflow is known as a non-trivial task (nondeterministic task or task not quick and easy to accomplish) [74]. However, apart from determination done using time consuming approaches like greedy and path-based determination, Ndamlabin et al. [73, 14, 75] proposed analytical determination that proved it effectiveness, and has the advantage of being fast. The aim of the algorithm is to find an optimal VM configuration that will aid to conserve energy in the could data center. To select the number of VM we use the
optimal number of VMs (optNbVMs) introduced in equation 4 [73]

$$optNbVMs = \left\{ \begin{array}{c} AvgW, if\, AvgW \leq StdDW \\ min\{AvgW + StdDW, MaxW\}, otherwise \end{array} \right\} \quad 4$$

where MaxW is the maximum, AvgW is the average and StdDW is the standard deviation of the levels' width of the workflow

Algorithm 1: ON-VM-DA

Input : K ⟶ set of VMs in a VM pool
Output:
CC-VM-G ⟶ current VM Configuration
OptimalVm = get⟶ n ⟶ VMconfig
foreach *vm in vmAllocation.keyset* **do**
  Access the workload
  queue tasks based on DL
  obtain number of VMs in the pool ⟶ eq2
  get UPC capacity of each VM ⟶ eq 3
  if *set of VM on CPU = workload$_{tn}$* **then**
    OptimalVM = OptimalVMconfig;
  end
  return OptimalVM
end

4.1.2. Determining the total CPU capacity needed

The aim of the algorithm is to schedule workflow tasks on cloud resources by determining the most cost optimal execution time without violating user deadlines. Therefore, we consider the total CPU capacity as the cost-optimal one under user deadline. To determine it, we consider the expected completion time in equation 5 and the makespan estimation proposed by [14]. Giving an instance type of VM denoted VMitk 290 and a DAG of tasks t1,. . . , tn , the estimated makespan is determined by using equation 4: Expected Completion Time of workflow task wti on vmk is denoted as ECT(wti , vmk), which can be computed by using equation 5. [37].

$$ECT = ET + loadvmk \quad (5)$$

Where ET is execution time of workflow task wti on vmk and $load_{vmk}$ is the workload of vmk at a given time.

$$estimate\mathbf{M}_G^k = \frac{\sum_{tiWT}\{ET(i,k) + max_{tj}\epsilon Succ_{(ti)\{TT(i,j)\}}\}}{optNbVMs} \quad 6$$

Where ET(ti; k) is the execution time of the task ti on a VM of type VM itk, and TT(i; j) is the transfer time of data from task ti to task tj, and optNbVMs is the number of VMs to be used for the execution of the whole workflow. By this, total capacity is estimated by equation 7, and total CPU is calculated using equation 8.

$$Total\ capacity = estimate\mathbf{M}_G^k \times optNbVMs \quad (7)$$

$$Total\ CPU = \frac{\sum_{tiWT}\{ET(i,k) + max_{tj}\epsilon Succ_{(ti)\{TT(i,j)\}}\}}{optNbVMs} \quad (8)$$

Where VMitk is the cheapest instance type representing equation 9.

$$totalCPU \leq \delta \times optNbVMs \quad (9)$$

Where δ is the user deadline. Let us denote that VMitk as $VMit^{opt}$.

Deadline DL: Deadlines are always specified or defined by the users. The goal of HPC users are that, every task in a workflow should be executed entirely and the feedback (results) are communicated back to the user indicating that the tasks scheduling process has completed successfully and that, all tasks were scheduled on/before their users specified deadline. If the task execution time exceeds the user specified deadline, then, QoS constraint is violated [76]. Since all the tasks are assumed to have deadlines, deadline queue of workflow tasks is established on the arrival of the tasks

4.1.3. Determining the optimal VM configuration (number and types of VMs) Here we have to determine several sets of VMs, such that in each set the number of VMs is optNbVMs, and the sum of their capacity (almost) equal to total CPU. We could have several VMs of the same type in a set. To do that, we determine first the standard deviation of power among the ones proposed by the provider in condition 10 Also, we consider two set of VMs types which is presented in 11 and 12.

$$PwStdD = stdD\{pk\}\ |V\ M\ it_k\ in\ VMIT \quad (10)$$

Let consider two set of VM types

$$S1 = \{VMit_k \ \ in \ \ VMIT \mid Pvmit^{opt} - PwStdD \leq p_k \leq Pvmit^{opt}\} \quad (11)$$

$$S2 = \{VMit_k \ \ in \ \ VMIT \mid Pvmit^{opt} \leq p_k \leq Pvmit^{opt} + PwStdD\} \quad (12)$$

Take alternatively VM of type in S2 and in S1 until we have optNbVMs VMs. All the VMs must be distinct, but two VMs can be of same type. That means, we have enough VMs of different types s available.

Algorithm 2: MOWOS

Input : Given a DAG, G=(W,E), where w has n workflow tasks, tw_3,.....wt_n)
Output:

$wt_{queue\text{-}list} = (wt_1, wt_2, tw_3, ...wt_n)$
while $wt_{queue} \neq \phi$ **do**
  compute ECT of each wt using equation 3
  Create Dl_{queue} and queue the tasks based on their deadlines
  foreach *ECT of wt_i on vm_k* **do**
    Compare the ECT of wt_i on vm_k to its deadlines
    if *the ECT_{wti} <= Dl_{wti}* **then**
      generate mapping for wti
    else
      if *ECT_{wti} > Dl_{wti}*
      allow MOW-PSO to produce mapping
      Update Dl_{queue}
      while *$Dl_{queue} \neq \phi$* **do**
        Repeat step 7 to step 14
      end
      if *Dl_{queue} is empty*
    end
  end
end

Using MOWOS to produce mappings, and PSO to initialize the mappings produced MOWOS A multi-objective workflow Optimization strategy (MOWOS) that optimizes workflow tasks execution is applied to execute workflows using the

configuration of VMs determined in the previous step (step 3). MOWOS uses Task Splitting method, Minimum VM (MinVM) selection and Maximum VM (MaxVM) selection strategies to ensure that a cost-optimized VM is selected for a data-intensive workflow application. Considering that the workflow have n tasks, and that our determined configuration have p VMs, we denote the mapping as following:

4.2. Using MOWOS to produce mappings, and PSO to initialize the mappings produced MOWOS A multi-objective workflow Optimization strategy (MOWOS) that optimizes workflow tasks execution is applied to execute workflows using the configuration of VMs determined in the previous step (step 3). MOWOS uses Task Splitting method, Minimum VM (MinVM) selection and Maximum VM (MaxVM) selection strategies to ensure that a cost-optimized VM is selected for a data-intensive workflow application. Considering that the workflow has n tasks, and that our determined configuration has p VMs, we denote the mapping as following:

T_1	t_2	……..	t_i	……	t_n
Map(t_1)	Map(t_2)	……..	Map(t_i)	……	Map(t_n)

Map(ti) is the VM onto which MOWOS has mapped task (ti).

After we applied MOWOS to execute workflow tasks per the optimized VMs configuration, we use PSO to initialize the mappings produced by MOWOS. PSO is a nature-inspired algorithm that mimics the behavior of a fock of birds in a search for a better food source (solution). We consider a population of m Particles, each being a potential solution to the scheduling problem. We then assign the mapping of MOWOS unto the first Particle. For the other ones, we keep the order of execution of tasks as obtained from MOWOS, but for the VMs, we regenerated them as in step 3 of the proposed MOW-PSO Model. This allows us to keep the same mapping affinity between tasks (task mapped onto same VM). During the process, we ensured that there is no doubloon. Therefore, the initial population may have less than m Particles. To complete the hybridization process, we finally initialized the Global solution to the mapping obtained by MOWOS.

Algorithm 3: MOW-PSO Algorithm

Input : Given a DAG, G=(W,E), where w has n workflow tasks (wt_1, wt_2, tw_3,.....wt_n)

Output:

Given a set of VMs and set of Workflow tasks
Initialize the particles position as produced by MOWAS
For each particle I € N
→*Pbest* = $p(Wt1, \ldots\ldots wtn)$ =Determine the Pbest with the current position of the particle
→*Gbest* =g(among all wt's) = Determine the Gbest with the best position of all the particles
Fitness = f(VM1 ……vmK)
if *its fitness is less than its pbest, then pbest is set as the current value and location* **then**
|
else
end

5. EVALUATION SETTING AND PERFORMANCE EVALUATION

This section detailed the simulations settings and provided a set of experimental results to observe the performance of the proposed MOW-PSO, a hybrid-based multi-criteria approach with the state-of-the-art algorithms such as REEWS[77], MS PSO [78] and MOWOS [37].

5.1 Datasets Used

In a real world simulation platform, the core operational function is to get a proper data set that will help for the simulation and bench-marking of the algorithms. In this research, we use five real-world workflow applications (Montage, CyberShake, SIPHT, and LIGO) during the experimentation and bench-marking. The Workflows (data set) was obtained from the Pegasus Workflow repository (https://confluence.pegasus.isi.edu/display/pegasus/Deprecated+Workflow+Generator). The montage [79] is for astronomy applications created by the NASA/IPAC Infrared Science Archive [80]. It is an open-source toolkit that is used in the construction of large image mosaics of the sky based on a set of input images. It is a data-intensive workflow application that requires a lot of time for data transfer. CyberShake [81] is a seismology workflow application used for the computation of Probabilistic Seismic Hazard curves in earthquake research to characterize the earthquake hazard in a region. LIGO [82], is for astrophysics science for the detection of gravitational waves. It is CPU intensive application that requires much memory for processing. SIPHT [82] workflow application is applied in the field of bioinformatics science which involves more complex data of various sizes requiring a higher system with a higher processing capacity for processing. The Structure of the various workflow applications is presented in Figure 6. A summary of the data set is broken into four scenarios for each workflow application such as small, medium, large and extra-large workloads and is summarized in Table 1, which is similar to the settings in [32, 34, 83].

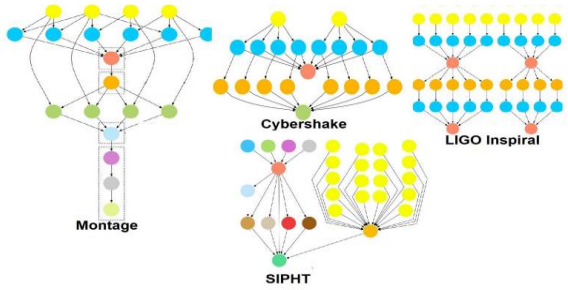

Fig. 6: Workflow width distribution [73]

Table 1: Workflow Data Set Distribution.

Scenarios	Workflows	Number of Tasks	Description
1	Montage	100, 300, 500 and 1000	Small, Medium, Large and Extra-large Workload
2	Inspiral	100, 300, 500 and 1000	Small, Medium, Large and Extra-large Workload
3	CyberShake	100, 300, 500 and 1000	Small, Medium, Large and Extra-large Workload
4	SIPHT	100, 300, 500 and 1000	Small, Medium, Large and Extra-large Workload

5.2 Experimental setup

This section detailed both the hardware and software configuration for the simulations. The hardware configuration is done using MacBook Air A1932, Intel(R) Core(TM) i5-8210Y, CPU @3.2GHz, 8 GB memory. The software environments used are Windows 10, Eclipse 3.5 and WorkflowSim-1.0. WorkflowSim is a Java-based simulation toolkit used for modeling and evaluating scheduling algorithms to ascertain their effectiveness. It provides high-level abstractions for cloud computing components, supports resource provisioning and scheduling policies, and has been widely adopted for research and education [84]. The inputs to the simulation environment are as follows: 1) the average bandwidth between resources is 20 MBps as in [65, 75], which is the average bandwidth setting offered by Amazon

Web Services [85, 86], 2) the processing matrix for each VM is measured in Million Instruction Per Second (MIPS) as in [19, 25, 87], 3) the task lengths are set in Million Instruction (MI) as in [25].

6. RESULTS AND DISCUSSIONS

To exhibit the performance of the algorithms, we based our analyzes on the following four (4) dimensions: First, we evaluated the results based on energy consumption using the number of workflow tasks executed. Secondly, we performed the evaluation on the basis of tasks to VM assignment cost, thirdly, we performed the evaluation based on the execution makespan and finally, we evaluated our solution based on load balancing efficiency. The results for each of the evaluation criteria are presented below:

6.1 Evaluation based on Energy Consumption

To evaluate the energy consumption of the three (3) solutions, we depicted the energy consumption for each of the four workflows used in Figures. 7, 8, 9 and 10. Results for the Montage workflow application is plotted in Figure 9. As can be seen, the proposed MOW-PSO is superior in conserving energy in all the workflow tasks sizes compared to the REEWS and MS PSO algorithms. However, when the number of tasks increases, the energy consumption for all the algorithms also increased. This is a reflection that the energy consumption of a data center is directly proportional to the number of tasks executed over a period. The higher the number of tasks processed in a data center, the higher the energy consumed. As the workload increases, the proposed MOW-PSO scheduler maintains its stand in conserving energy but in a decreasing rate and finally lost it to REEWS when the workflow tasks double up from 500 to 1000. This observation provides evidence that, with a smaller, medium and larger workload (100, 300 and 500) the proposed scheduler tends to be more energy efficient, however, with extra-large workloads, this efficiency decreases.

In Figure 8, the result of CyberShake workflow is observed. The results show that MOW-PSO scheduler presents better energy efficiency in almost all the four tasks sizes compared to REEWS and MS PSO. However, as the workload (number of tasks) reduced to 100, the energy efficiency for the proposed scheduler decreases. The results show that for small workload of 100, REEWS conserves energy significantly better than the proposed MOW-PSO and the MS PSO algorithms. This is because at workflow tasks 100, REEWS algorithm achieves better resource allocation and therefore able to conserve energy better than the MOWPSO algorithm.

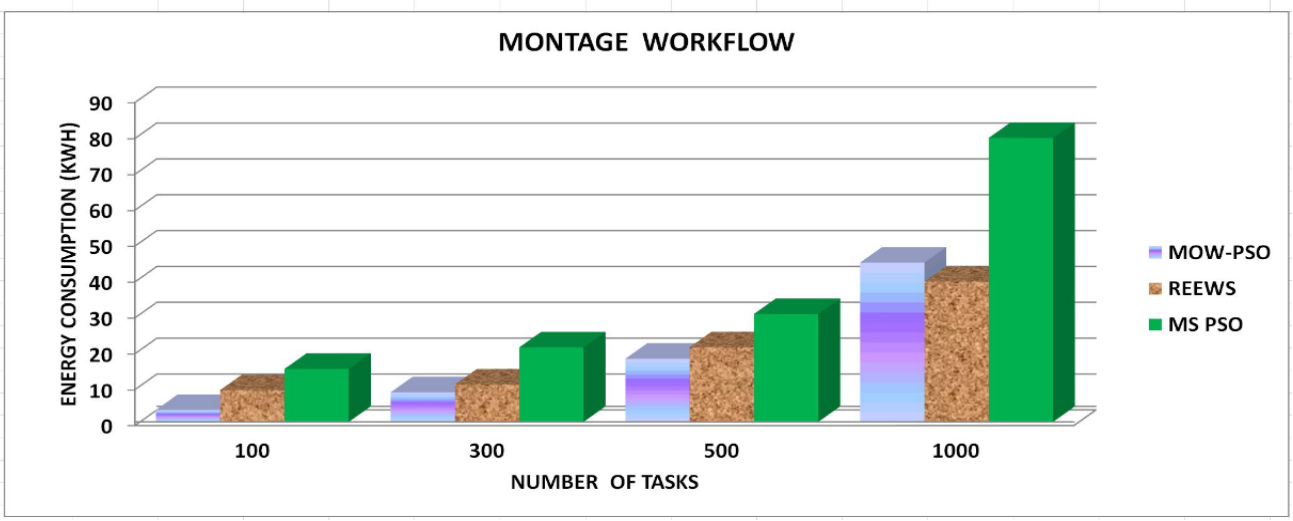

Figure 7: Montage-comparison Experiments in Energy Consumption

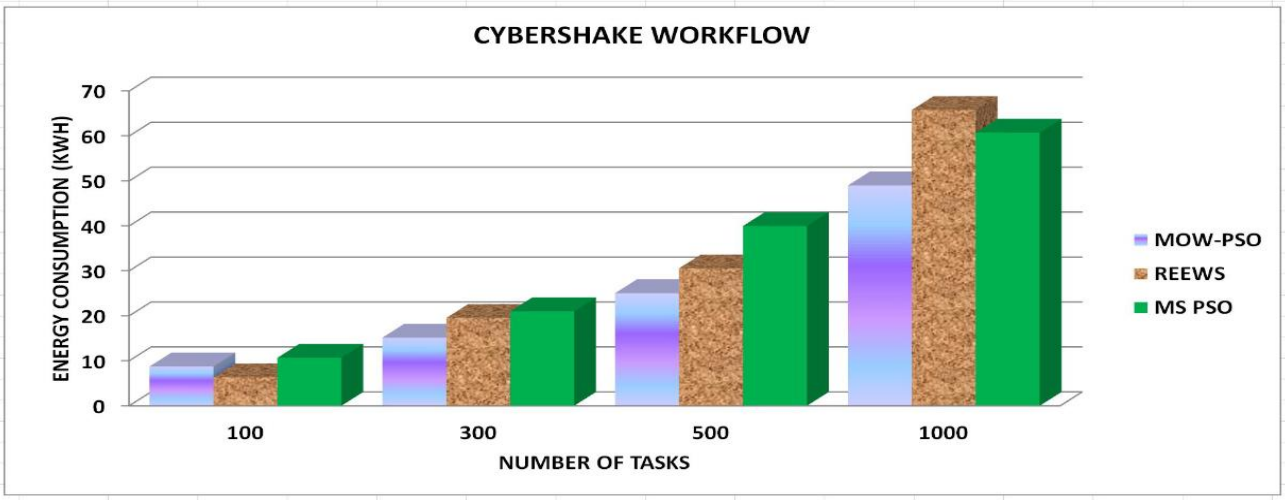

Figure 8: CyberShake-Comparison Experiments in Energy Consumption

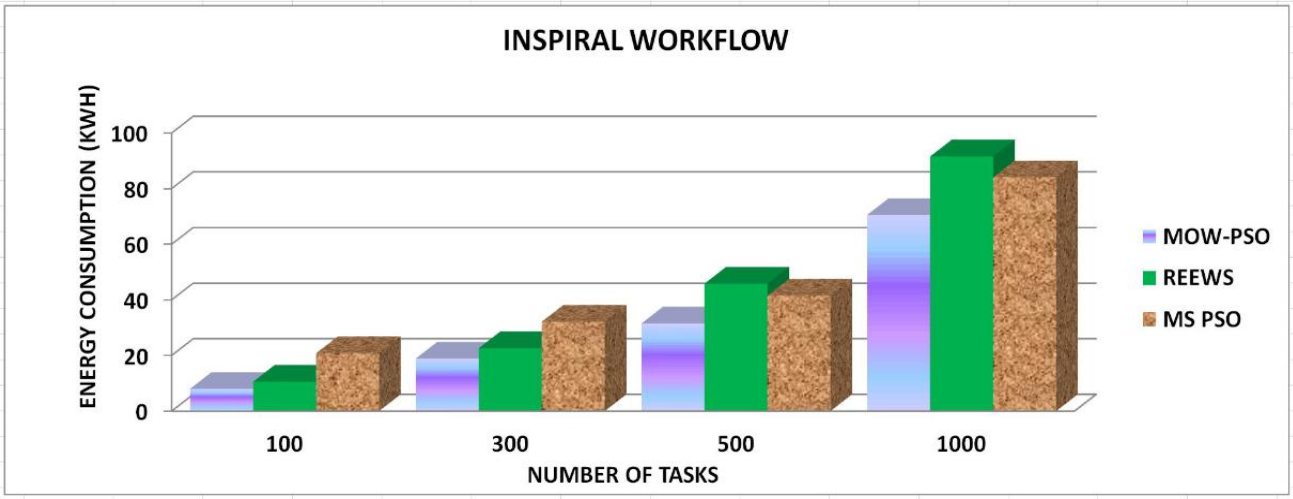

Figure 9: Inspiral-Comparison Experiments in Energy Consumption

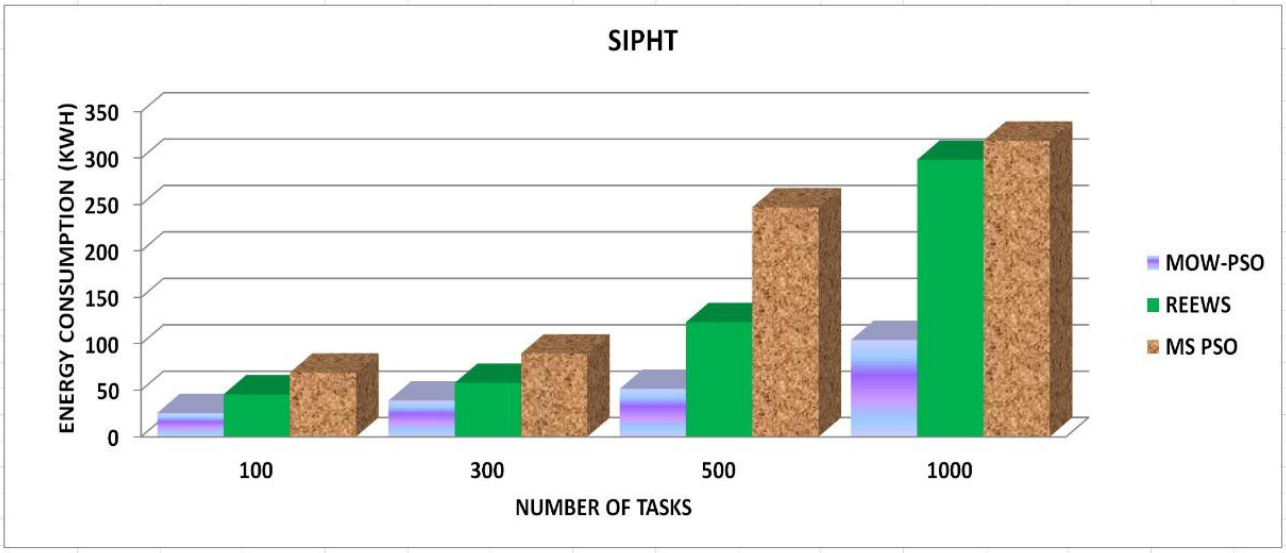

Figure 10: Sipht-Comparison Experiments in Energy Consumption

6.2. Evaluation based on Cost

We present the results of execution cost obtained by the proposed MOW-PSO, MOWOS, REEWS and MS PSO schedulers by using Montage, CyberShake, LIGO Inspiral and SIPHT workflow applications of various asks sizes. Figures 11–14 shows the plotted bar graphs for the four workflows with different load sizes used to compare between the MOW-PSO, MOWOS, REEWS and MS PSO. From the figures, we can observe that he proposed MOW-PSO produced cheaper schedules compared to the MOWOS, REEWS and MS PSO for

almost all the workflow categories, such as small, medium, large and extra-large tasks. However, the proposed algorithm failed to generate cheaper schedules with 100 tasks of CyberShake and 1000 tasks of SIPHT as MS PSO and MOWOS dominated by generating cheaper schedules in CyberShake of 100 and SIPHT of 1000 workflow tasks respectively. Notwithstanding, it is visible that the performance of the proposed MOW-SPO is better than the MOWOS, REEWS and MS PSO for any of the aforementioned workflow applications.

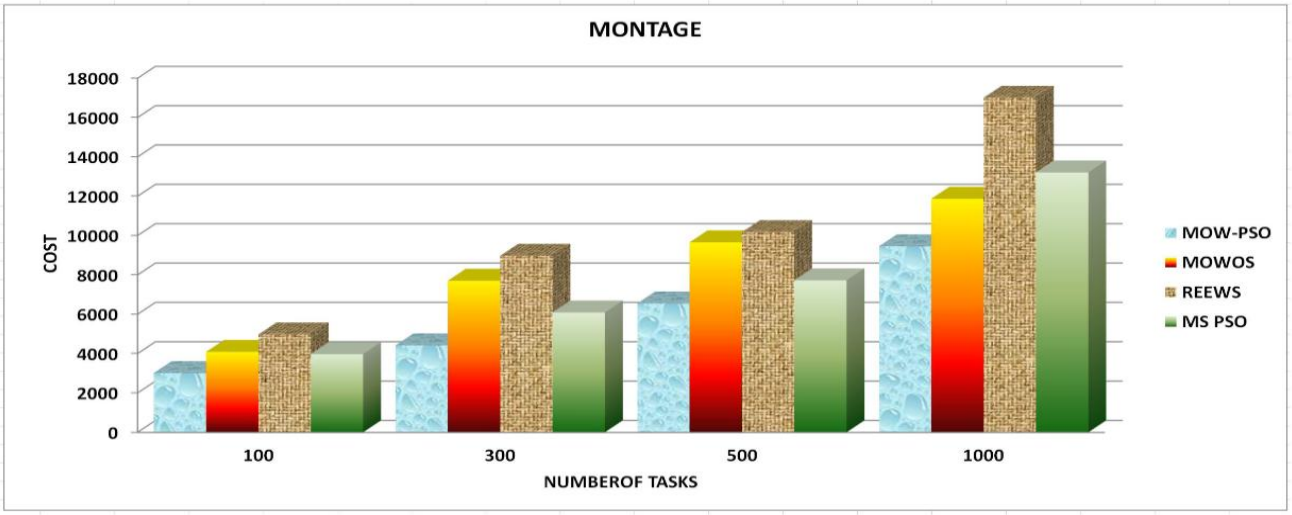

Figure 11: Montage-comparison Experiments in Execution Cost

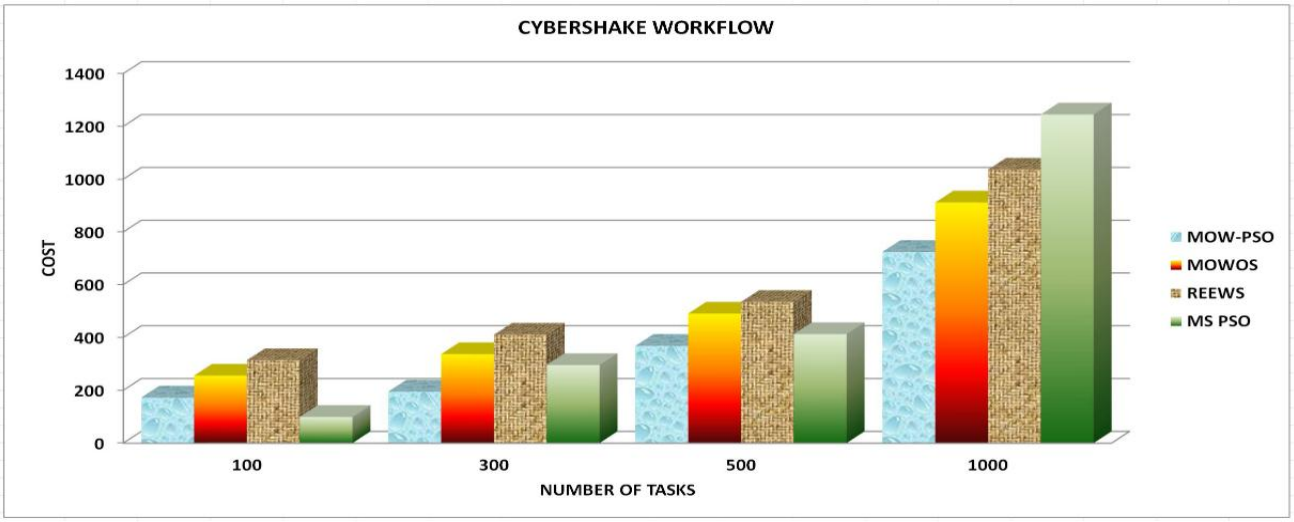

Figure 12: CyberShake-Comparison Experiments in Execution Cost

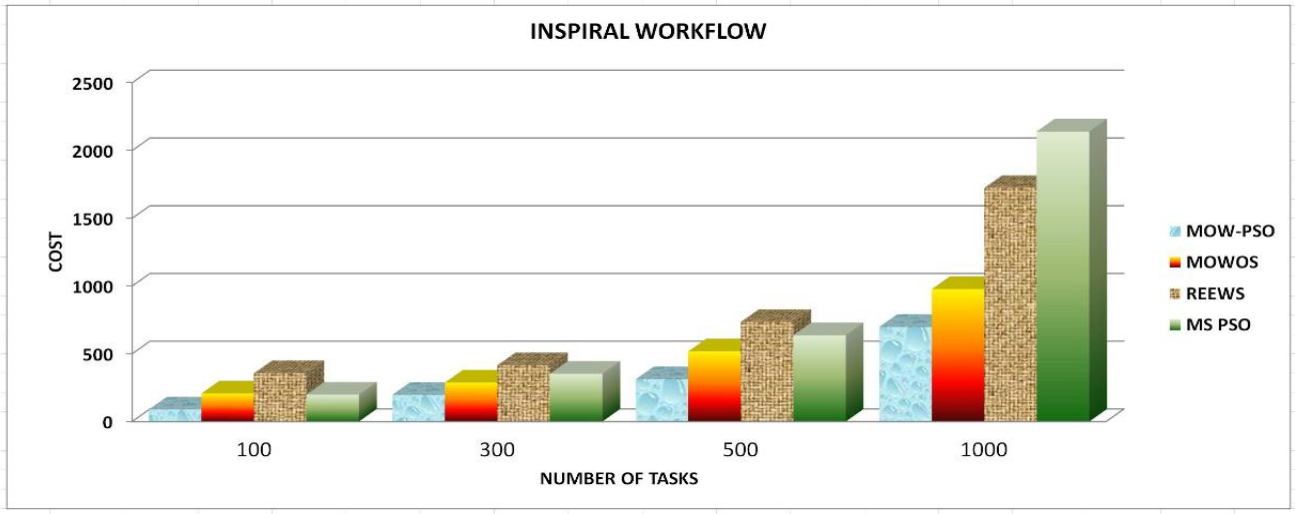

Figure 13: Inspiral-Comparison Experiments in Execution Cost

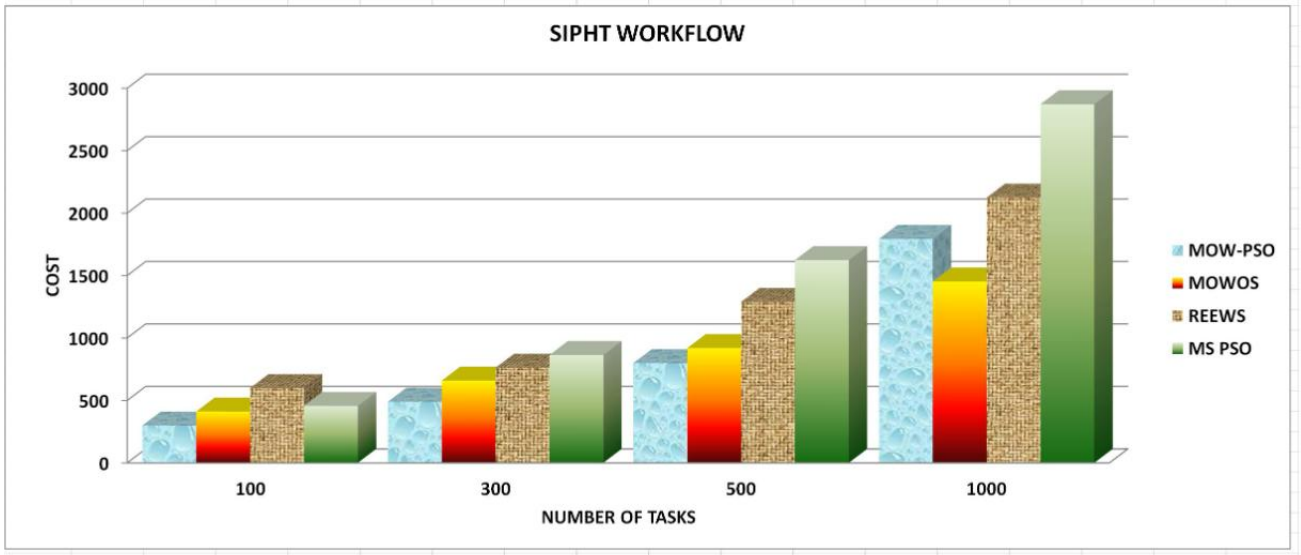

Figure 14: Sipht-Comparison Experiments in Execution Cost

6.3. Evaluation Based on Makespan

2. This subsection shows the performance of makespan under four different workflows with different tasks sizes of 100, 300, 500 and 1000 for each workflow application. In Figure 15, we show the performance of the algorithms with respect to the Montage workflow application. The proposed MOW-PSO generates shortest schedules from all the four different job sizes. For example, at job 100, all the algorithms exhibit a similar performance. At tasks 500 and 1000, the REEWS algorithm exhibits the worst performance, while the MOW-PSO algorithm generated the shorter schedules, followed by MOWOS and MS PSO.

The next round of the experiments demonstrates the performance of the four algorithms in generating a schedule within a specified time (makespan) using the CyberShake workflow application as in Figure 16. The REEWS algorithm exhibited the worst performance at all the job sizes followed by MS PSO. The MOWOS algorithm is able to generate shorter schedules at job 100 and 300 better than MS PSO. The MOW-PSO is providing the best solution that generates cheaper schedules with CyberShake no matter the job size. In the Inspiral workflow application which is plotted at Figure 17, the MS PSO algorithm has over-performed the proposed algorithm and produced shorter schedules at job 500 and 1000. This is because the proposed algorithm is hybrid, its uses the functions of both heuristics and meta-heuristics and therefore unable to balance the workload easily on cloud resources to generate cheaper schedules when large workflows of Inspiral is used. However, the proposed algorithm is efficient in generating cheaper schedules with Inspiral when using small tasks of 100 and medium tasks of 300. The REEWS performed worst at tasks 100, 300 and 1000, but able to generate cheaper schedules than MOWOS at tasks 500. Finally, in the SIPHT workflow plotted at Figure 18, REEWS produced the worst makespan at all the tasks sizes, followed by the MS PSO. The MOW-PSO achieve the best makespan values at all levels of SIPHT followed by the MOWOS algorithm.

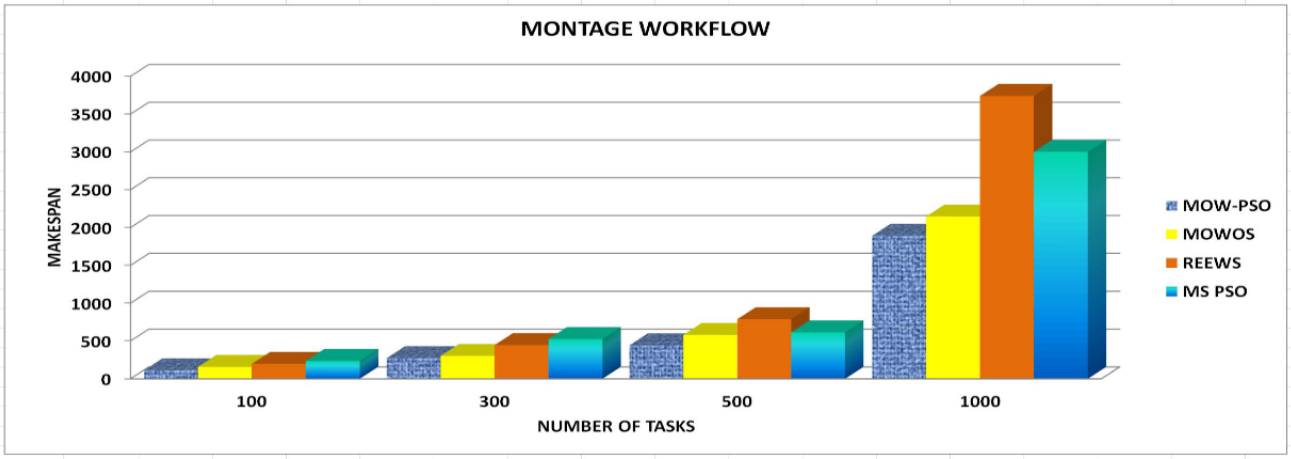

Figure 15: Montage-comparison Experiments in Makespan

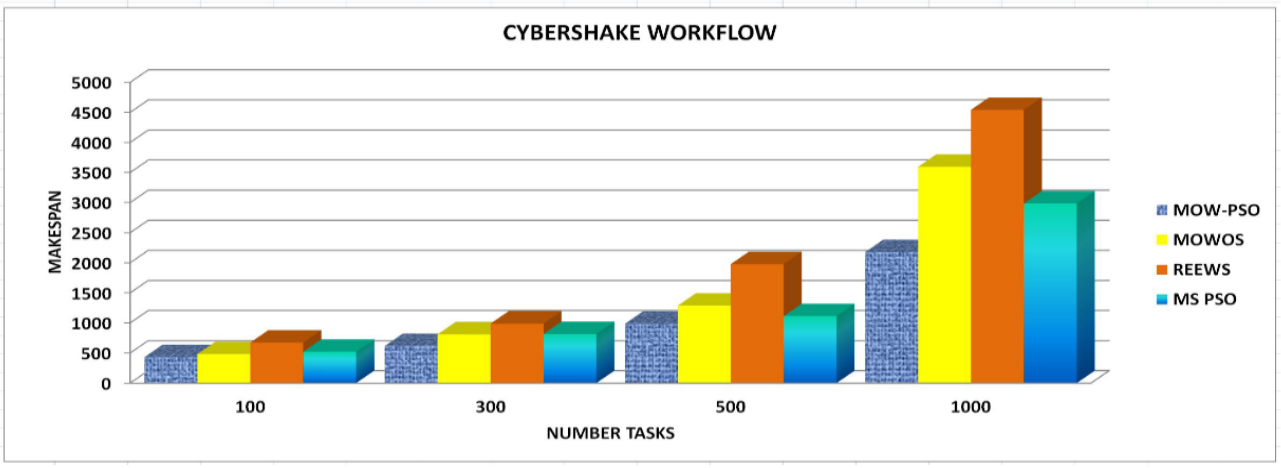

Figure 16: CyberShake-Comparison Experiments in Makespan

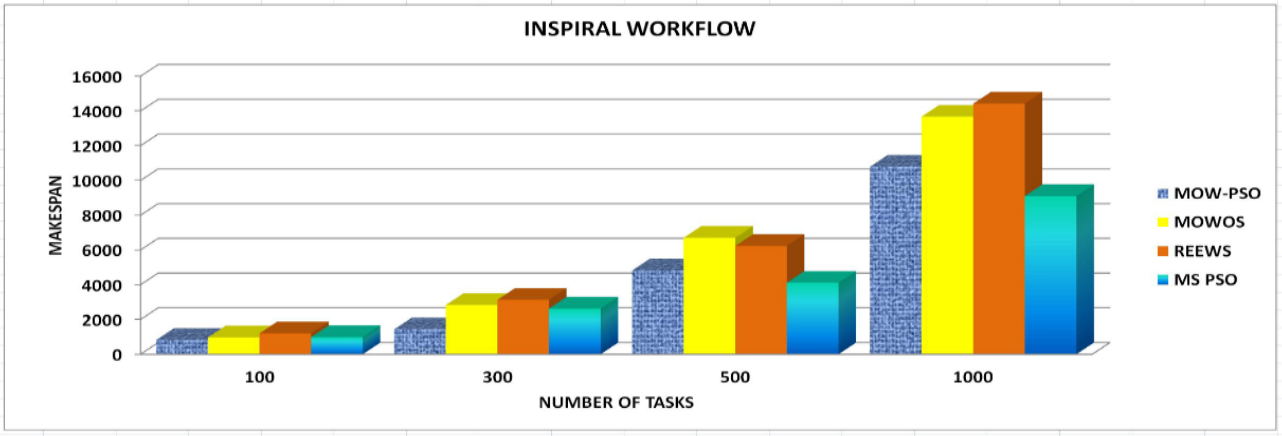

Figure 17: Inspiral-Comparison Experiments in Makespan

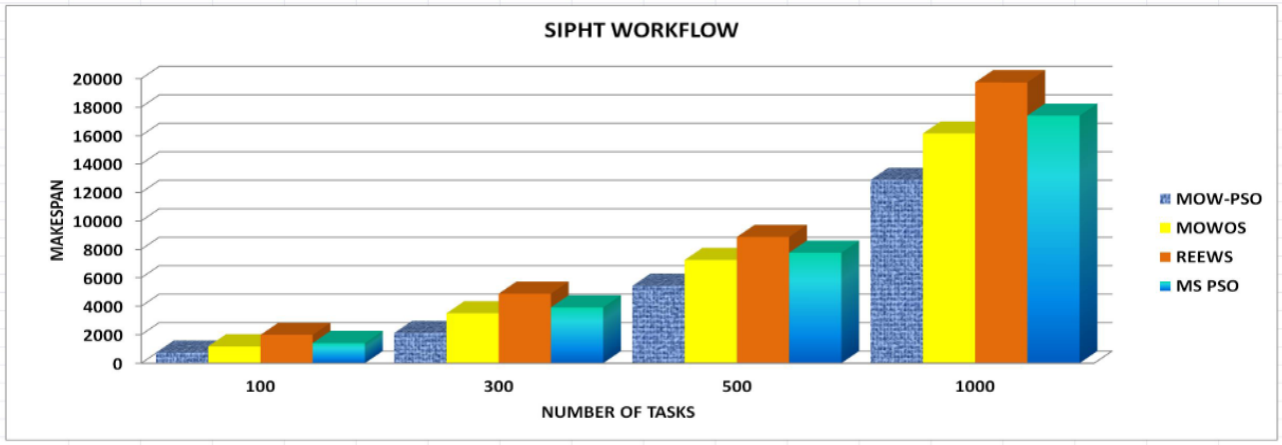

Figure 18: Sipht-Comparison Experiments in Makespan

6.4 Evaluation Based on Load Balancing

Having compared the algorithms based on energy consumption, execution cost, and execution makespan, we also analyzed the algorithm on load balancing efficiency. We plotted the results of load balancing efficiencies in Figures 19, 20, 21 and 22 using Montage, CyberShake, Inspiral and SIPHT workflows with various load sizes such as 100, 300, 500 and 1000 across all the four workflows. Overall, the proposed MOW-PSO algorithm adjudged the best load balancer on all the instances, from small size workload of 100 to extra-large workload of 1000. For example, the results in figure 19 projects the WOW-PSO as the best performer on 100 workloads on Montage, CyberShake, Inspiral and SIHPT workflows, while the MS PSO placed second on workload of 100 load of Inspiral and 100 of SIPHT, however, PS PSO lost the second position to MOWOS on 100 workloads of Montage and to REEWS on 100 workloads of CyberShake. In Figure 20, similar load balancing performance is recorded except in Inspiral that the MS PSO produced good load balancing results than the proposed algorithm. Also, Figures 21, we have observed a different pattern of the results across the four workflows where each algorithm produced lesser load balancing efficiency except in CyberShake. Moreover, the proposed algorithm

failed to balance workload on cloud resource when Inspiral of 500 tasks are scheduled. For 1000 workload as plotted in Figure 22, solutions from MOW-PSO meet all the load distribution conditions and therefore able to distribute workflow evenly across all the four workflow better than MOWOS, REEWS and MS PSO. However, it efficiencies reduces when CyberShake, Inspiral and SIPHT workflows are used.

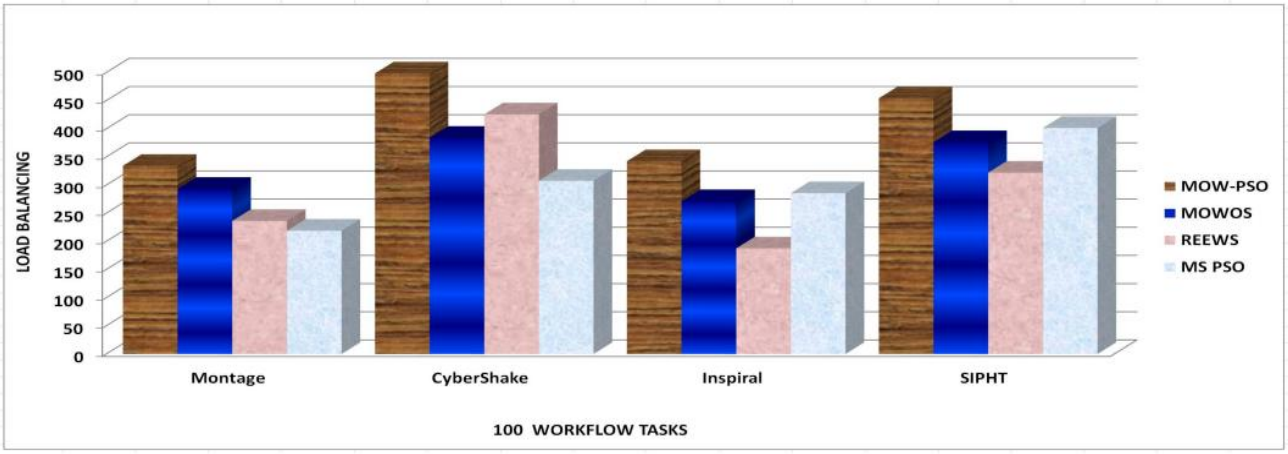

Figure 19: Comparison of Load Balancing using 100 Tasks

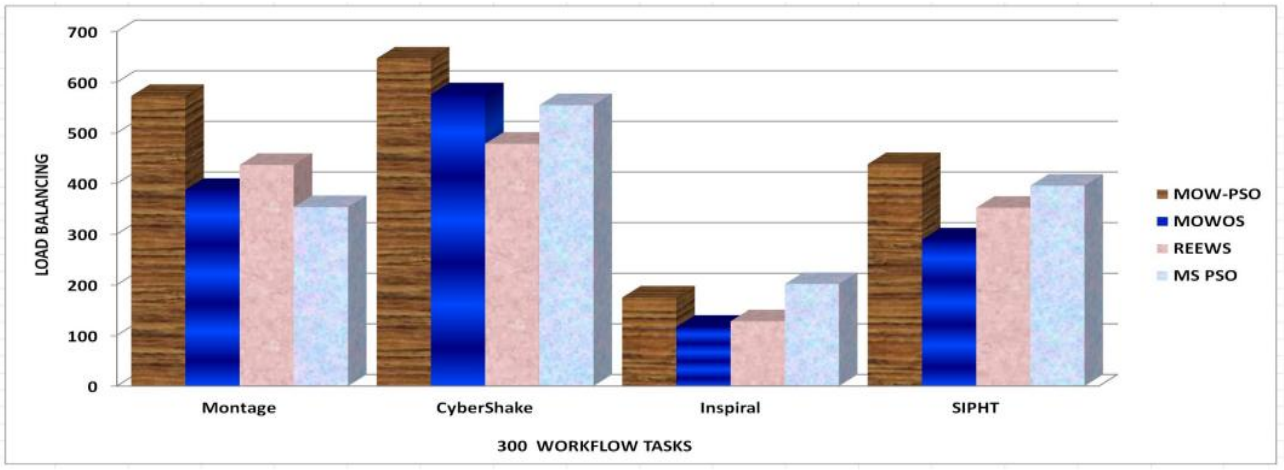

Figure 20: Comparison of Load Balancing using 300 Tasks

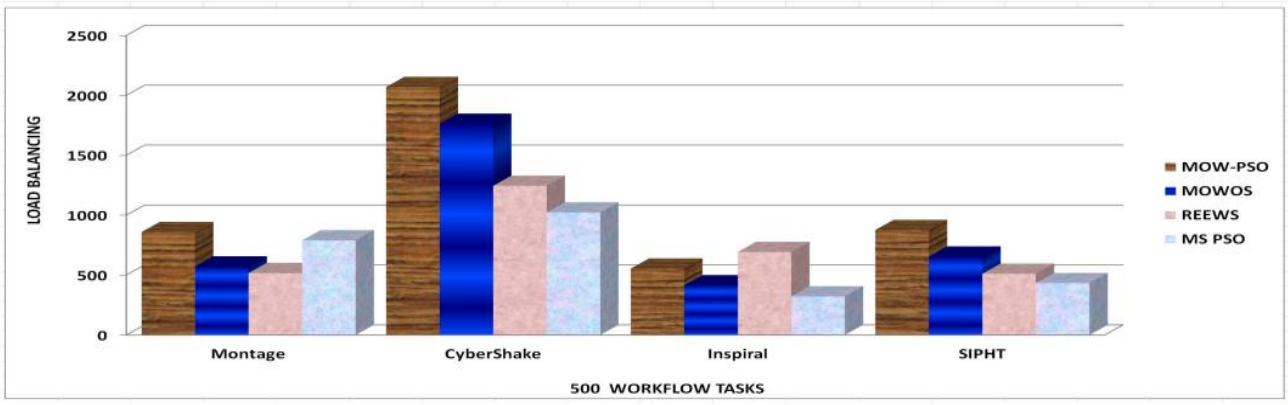

Figure 21: comparison of Load Balancing using 500 Tasks

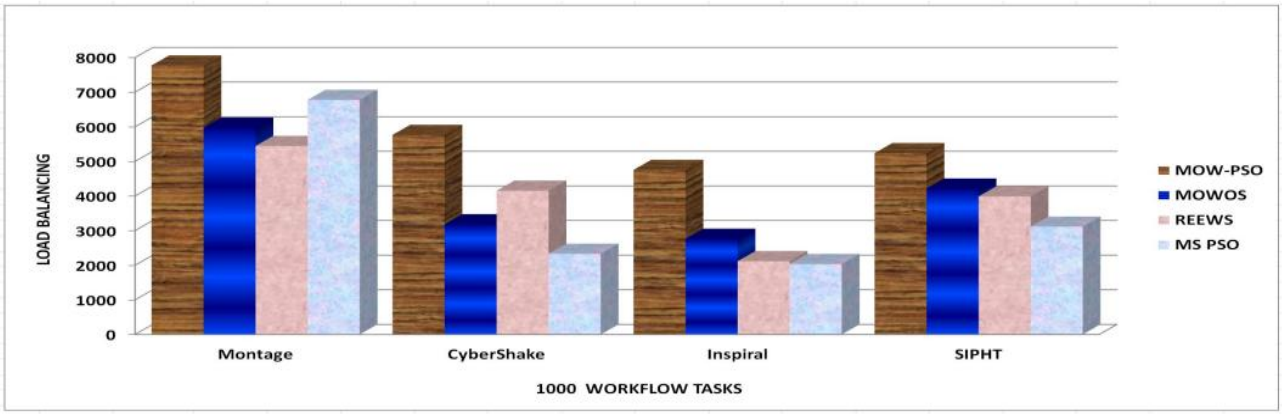

Fig 22: comparison of Load Balancing using 1000 Tasks

7. CONCLUSION AND FUTURE WORK

In this paper, we analyzed the impact of the various workflow optimization methodologies and presented a formal definition of the workflow optimization problem (VM provisioning) and its influence over execution cost, execution makespan, and energy consumption during workflow scheduling. We then designed a hybrid scheduling scheme (MOW-PSO) based on heuristic and meta-heuristic algorithms that orchestrate well with cloud resources to determine the most cost-optimal VM configuring. The MOW-PSO algorithm strives to minimize energy consumption, execution cost and execution makespan without violating user-defined budget and deadline constraints. Our proposed method introduces a new concept that is responsible for determining the optimal number of instants that can be configured on CPU to speed up execution time of workflow tasks. This is done to avoid resource wastage at the expense of the user. Through comparative experiment, we can conclude that, the proposed approach is efficient in providing a good result in energy consumption, execution cost and makespan against the state-of-the-art algorithms. However, our work is implemented to consider static VMs provisioning. Since cloud computing operates within a dynamic environment, our future work will apply DVFS technique for a dynamic VM provisioning in a could data center.

8. REFERENCES

[1] J. K. Konjaang, Optimizing workflow scheduling for high-performance computing, Ph.D. thesis, Univer sity College Dublin. School of Computer Science (2025).

[2] G. Di Luna, R. Baldoni, Non trivial computations in anonymous dynamic networks, in: 19th International Conference on Principles of Distributed Systems (OPODIS 2015), Schloss Dagstuhl-LeibnizZentrum fuer Informatik, 2016.

[3] F. Kuhn, R. Oshman, Dynamic networks: models and algorithms, ACM SIGACT News 42 (1) (2011) 465 82–96.

[4] A. Casteigts, P. Flocchini, W. Quattrociocchi, N. Santoro, Time-varying graphs and dynamic networks, International Journal of Parallel, Emergent and Distributed Systems 27 (5) (2012).

[5] W. Ahmad, B. Alam, S. Ahuja, S. Malik, A dynamic vm provisioning and de-provisioning based costefficient deadline-aware scheduling algorithm for big data workflow applications in a cloud environment, Cluster Computing 24 (1) (2021) 249– 278.

[6] M. A. Rodriguez, R. Buyya, Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, IEEE transactions on cloud computing 2 (2) (2014) 222–235.

[7] M. Mao, M. Humphrey, Auto-scaling to minimize cost and meet application deadlines in cloud workflows, in: SC'11: Proceedings of 2011 International Conference for High Performance Computing, Networking, and Analysis, IEEE, 2011, pp. 1–12.

[8] J. Ndamlabin Mboula, V. Kamla, M. Hilman, C. Tayou Djamegni, Energy-efficient workflow scheduling based on workflow structures under deadline and budget constraints in the cloud, arXiv e-prints (2022) arXiv–2201.

[9] Casas, J. Taheri, R. Ranjan, L. Wang, A. Y. Zomaya, A balanced scheduler with data reuse and replication for scientific workflows in cloud computing Future Generation Computer Systems 74 (2017) 168–178

[10] G. Singh, C. Kesselman, E. Deelman, Application- level resource provisioning on the grid, in: 2006 Secon IEEE International Conference on e-Science and Grid Computing (e-Science'06), IEEE, 2006, pp. 83–83.

[11] S. Chaisiri, B.-S. Lee, D. Niyato, Optimization of resource provisioning cost in cloud computing, IEEE transactions on services Computing 5 (2) (2011) 164–177

[12] M. Al-Ayyoub, Y. Jararweh, M. Daraghmeh, Q. Althebyan, Multi-agent based dynamic resource pro visioning and monitoring for cloud computing systems infrastructure, Cluster Computing 18 (2) (2015) 919–932.

[13] X. Li, Z. Cai, Elastic resource provisioning for cloud workflow applications, IEEE Transactions on Automation Science and Engineering 14 (2) (2015) 1195–1210

[14] J. E. Ndamlabin Mboula, V. C. Kamla, C. Tayou Djam´egni, Dynamic provisioning with structure inspired selection and limitation of vms based cost- time efficient workflow scheduling in the cloud, Cluster Computing 24 (3) (2021) 2697–2721.

[15] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, in: Advances in computers, Vol. 82, Elsevier, 2011, pp. 47–111.495

[16] R. Buyya, Introduction to the ieee transactions on cloud computing, IEEE Transactions on cloud computing 1 (1) (2013) 3–21.

[17] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, M. A. Kozuch, Heterogeneity and dynamicity of clouds at scale: Google trace analysis, in: Proceedings of the third ACM symposium on cloud computing, 2012, pp. 1–13.500

[18] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41 (1) (2011) 23– 50.

[19] M. A. Rodriguez, R. Buyya, Scheduling dynamic workloads in multi-tenant scientific workflow as a service platforms, Future Generation Computer Systems 79 (2018) 739–750.

[20] E. Deelman, G. Singh, M. Livny, B. Berriman, J. Good, The cost of doing science on the cloud: the montage example, in: SC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing, Ieee, 2008, pp. 1–12.

[21] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, N. J. Wright, Performance analysis of high performance computing applications on the amazon web services cloud, in: 2010 IEEE second international conference on cloud computing technology and science, IEEE, 2010, pp. 159–168.

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

[23] R. Madduri, K. Chard, R. Chard, L. Lacinski, A. Rodriguez, D. Sulakhe, D. Kelly, U. Dave, I. Foster, The globus galaxies platform: delivering science gateways as a service, Concurrency and Computation: Practice and Experience 27 (16) (2015) 4344–4360.

[24] J. K. Konjaang, J. Murphy, L. Murphy, Energy- efficient virtual-machine mapping algorithm (evima) for workflow tasks with deadlines in a cloud environment, Journal of Network and Computer Applications 203 (2022) 103400.

[25] V. Singh, I. Gupta, P. K. Jana, An energy efficient algorithm for workflow scheduling in iaas cloud, Journal of Grid Computing 18 (3) (2020) 357–376.

[26] J. Hartmanis, Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson), Siam Review 24 (1) (1982) 90

[27] S. Abrishami, M. Naghibzadeh, D. H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future generation computer systems 29 (1) (2013) 158–169.

[28] X. Ye, S. Liu, Y. Yin, Y. Jin, User-oriented many-objective cloud workflow scheduling based on an improved knee point driven evolutionary algorithm, Knowledge-Based Systems 135 (2017) 113–124.

[29] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, R. Rastogi, et al., Load balancing of nodes in cloud using ant colony optimization, in: 2012 UKSim 14th international conference on computer modelling and simulation, IEEE, 2012, pp. 3–8.

[30] L. Singh, S. Singh, A genetic algorithm for scheduling workflow applications in unreliable cloud environment, in: International conference on security in computer networks and distributed systems, Springer, 2014, pp. 139–150.

[31] G. Yao, Y. Ding, L. Ren, K. Hao, L. Chen, An immune system-inspired rescheduling algorithm for workflow in cloud systems, Knowledge-Based Systems 99 (2016) 39–50.

[32] Y. C. Lee, H. Han, A. Y. Zomaya, M. Yousif, Resource-efficient workflow scheduling in clouds, Knowledge-Based Systems 80 (2015) 153–162.

[33] H. Zhang, J. Shi, B. Deng, G. Jia, G. Han, L. Shu, Mcte: minimizes task completion time and execution cost to optimize scheduling performance for smart grid cloud, IEEE Access 7 (2019) 134793– 134803.

[34] A. Choudhary, I. Gupta, V. Singh, P. K. Jana, A gsa based hybrid algorithm for bi-objective workflow scheduling in cloud computing, Future Generation Computer Systems 83 (2018) 14–26.

[35] J. Kakkottakath Valappil Thekkepuryil, D. P. Suseelan, P. M. Keerikkattil, An effective meta- heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment, Cluster Computing 24 (3) (2021) 2367–2384.545

[36] M. Kalra, S. Singh, Multi-objective energy aware scheduling of deadline constrained workflows in clouds using hybrid approach, Wireless Personal Communications 116 (3) (2021) 1743–1764.

[37] J. K. Konjaang, L. Xu, Multi-objective workflow optimization strategy (mowos) for cloud computing, Journal of Cloud Computing 10 (1) (2021) 1–19.

[38] Y. Xu, K. Li, L. He, L. Zhang, K. Li, A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems, IEEE Transactions on parallel and distributed systems 26 (12) (2014) 3208–3222.

[39] H. M. Fard, Multi-objective scheduling for scientific workflow applications in grid and cloud infrastructures, Ph.D. thesis, University of Innsbruck (2015).

[40] C. Vecchiola, R. N. Calheiros, D. Karunamoorthy, R. Buyya, Deadline-driven provisioning of resources for scientific applications in hybrid clouds with aneka, Future Generation Computer Systems 28 (1) (2012) 58–65.

[41] B. Javadi, J. Abawajy, R. O. Sinnott, Hybrid cloud resource provisioning policy in the presence of resource failures, in: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, IEEE, 2012, pp. 10–17.

[42] C. Li, L. Y. Li, Optimal resource provisioning for cloud computing environment, The Journal of Super computing 62 (2) (2012) 989–1022.

[43] S. He, L. Guo, Y. Guo, C. Wu, M. Ghanem, R. Han, Elastic application container: A lightweight approach for cloud resource provisioning, in: 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, IEEE, 2012, pp. 15–22.

[44] L. Zhang, Z. Li, C. Wu, Dynamic resource provisioning in cloud computing: A randomized auction approach, in: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, IEEE, 2014, pp. 33–441.

[45] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, T. Fahringer,

A multi-objective approach for workflow scheduling in heterogeneous environments, in: 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), IEEE, 2012, pp. 300–309.

[46] R. A. Haidri, C. P. Katti, P. C. Saxena, Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing, Journal of King Saud University- Computer and Information Sciences 32 (6) (2020) 666–683.

[47] F. Abazari, M. Analoui, H. Takabi, S. Fu, Mows: multi-objective workflow scheduling in cloud computing based on heuristic algorithm, Simulation Modelling Practice and Theory 93 (2019) 119–132.

[48] N. Rehani, R. Garg, Meta-heuristic based reliable and green workflow scheduling in cloud computing, International Journal of System Assurance Engineering and Management 9 (4) (2018) 811–820.

[49] H. Y. Shishido, J. C. Estrella, C. F. M. Toledo, M. S. Arantes, Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds, Computers & Electrical Engineering 69 (2018) 378–394.

[50] M. Melnik, T. Trofimenko, Polyrhythmic harmony search for workflow scheduling, Procedia Computer Science 66 (2015) 468–476.

[51] Z. Wu, Z. Ni, L. Gu, X. Liu, A revised discrete particle swarm optimization for cloud workflow scheduling, in: 2010 international conference on computational intelligence and security, IEEE, 2010, pp. 184–188.

[52] Z. Pooranian, M. Shojafar, J. H. Abawajy, A. Abraham, An efficient meta-heuristic algorithm for grid computing, Journal of Combinatorial Optimization 30 (3) (2015) 413–434.

[53] Z. Beheshti, S. M. H. Shamsuddin, A review of population-based meta-heuristic algorithms, Int. J. Adv. Soft Comput. Appl 5 (1) (2013) 1–35.

[54] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, M. J. Usman, Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment, PloS one 12 (5) (2017) e0176321.

[55] N. Soltani, B. Soleimani, B. Barekatain, Heuristic algorithms for task scheduling in cloud computing: A survey., International Journal of Computer Network & Information Security 9 (8).

[56] M. A. Rodriguez, R. Buyya, A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments, Concurrency and Computation: Practice and Experience 29 (8) (2017) e4041.

[57] G. Dhiman, Ssc: A hybrid nature-inspired meta- heuristic optimization algorithm for engineering applications, Knowledge-Based Systems 222 (2021) 106926.

[58] J. K. Konjaang, L. Xu, Meta-heuristic approaches for effective scheduling in infrastructure as a service cloud: a systematic review, Journal of Network and Systems Management 29 (2) (2021) 1–57.

[59] Y. Wu, A survey on population-based meta-heuristic algorithms for motion planning of aircraft, Swarm and Evolutionary Computation 62 (2021) 100844.

[60] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, A. El- Sayed, A new online scheduling approach for enhancing qos in cloud, Future Computing and Informatics Journal 3 (2) (2018) 424–435.

[61] X.-H. Hu, J.-C. Ouyang, Z.-H. Yang, Z.-H. Chen, An ipso algorithm for grid task scheduling based on satisfaction rate, in: 2009 International Conference on Intelligent Human-Machine Systems and Cybernetics, Vol. 1, IEEE, 2009, pp. 262–265.

[62] W.-N. Chen, J. Zhang, An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 39 (1) (2008) 29–43.

[63] J.-J. Liang, P. N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005., IEEE, 2005, pp. 124–129.

[64] A. M. Manasrah, H. Ba Ali, Workflow scheduling using hybrid ga-pso algorithm in cloud computing, Wireless Communications and Mobile Computing 2018.

[65] V. Arabnejad, K. Bubendorfer, B. Ng, Budget and deadline aware e-science workflow scheduling in clouds, IEEE Transactions on Parallel and Distributed systems 30 (1) (2018) 29–44.

[66] V. Singh, I. Gupta, P. K. Jana, A novel cost- efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources, Future Generation Computer Systems 79 (2018) 95–110.

[67] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, K. Vahi, Characterizing and profiling scientific workflows, Future generation computer systems 29 (3) (2013) 682–692.

[68] effective and low-complexity task scheduling for het erogeneous computing, IEEE transactions on parallel and distributed systems 13 (3) (2002) 260–274.

[69] S. Elmougy, S. Sarhan, M. Joundy, A novel hybrid of shortest job first and round robin with dynamic variable quantum time task scheduling technique, Journal of Cloud computing 6 (1) (2017) 1–12.

[70] M. H. Shirvani, A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems, Engineering Applications of Artificial Intelligence 90 (2020) 103501.

[71] N. Arora, R. K. Banyal, A particle grey wolf hybrid algorithm for workflow scheduling in cloud computing, Wireless Personal Communications 122 (4) (2022) 3313–3345.

[72] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, S. Hu, Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft, Future Generation Computer Systems 93 (2019) 278–289.

[73] J. Mboula, V. Kamla, M. Hilman, C. T. Djamegni, Energy-efficient workflow scheduling based on workflow structures under deadline and budget constraints in the cloud, arXiv preprint arXiv:2201.05429.

[74] O. Michail, I. Chatzigiannakis, P. G. Spirakis, Naming

and counting in anonymous unknown dynamic networks, in: Symposium on Self- Stabilizing Systems, Springer, 2013, pp. 281–295.

[75] J. E. N. Mboula, V. C. Kamla, C. T. Djamegni, Cost- time trade-off efficient workflow scheduling in cloud, Simulation Modelling Practice and Theory 103 (2020) 102107.

[76] W.-J. Wang, Y.-S. Chang, W.-T. Lo, Y.-K. Lee, Adaptive scheduling for parallel tasks with qos satis faction for hybrid cloud environments, The Journal of Supercomputing 66 (2013) 783–811.

[77] R. Garg, M. Mittal, L. H. Son, Reliability and energy efficient workflow scheduling in cloud environment, Cluster Computing 22 (4) (2019) 1283–1297.

[78] D. Subramoney, C. N. Nyirenda, Multi-swarm pso algorithm for static workflow scheduling in cloud- fog environments, IEEe Access 10 (2022) 117199– 117214.

[79] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, M.-H. Su, Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand, in: Optimizing scientific return for astronomy through information technologies, Vol. 5493, SPIE, 2004, pp. 221–232.

[80] S. Yassir, Z. Mostapha, T. Claude, Workflow scheduling issues and techniques in cloud computing: A systematic literature review, in: International Conference of Cloud Computing Technologies and Applications, Springer, 2017, pp. 241–263.

[81] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, et al., Cybershake: A physics- based seismic hazard model for southern california, Pure and Applied Geophysics 168 (3) (2011) 367– 381.

[82] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, K. Vahi, Characterization of scientific workflows, in: 2008 third workshop on workflows in support of large-scale science, IEEE, 2008, pp. 1–10.

[83] S. Saeedi, R. Khorsand, S. G. Bidgoli, M. Ramezanpour, Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing, Computers & Industrial Engineering 147 (2020) 106649.

[84] O. Ghandour, S. El Kafhali, M. Hanini, Computing resources scalability performance analysis in cloud computing data center, Journal of Grid Computing 21 (4) (2023) 61.

[85] M. R. Palankar, A. Iamnitchi, M. Ripeanu, S. Garfinkel, Amazon s3 for science grids: a viable solution?, in: Proceedings of the 2008 international workshop on Data-aware distributed computing, 2008, pp.55–64.

[86] J. Sahni, D. P. Vidyarthi, A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment, IEEE Transactions on Cloud Computing 6 (1) (2015) 2– 18.

[87] M. Adhikari, T. Amgoth, An intelligent water drops-based workflow scheduling for iaas cloud, Applied Soft Computing 77 (2019) 547–566.