Journal of Engineering and Applied Science

RESEARCH Open Access

Optimizing scientific workflow scheduling in cloud computing: a multi-level approach using whale optimization algorithm

Xiaowen Zhang[1*]

*Correspondence:
qtianmumu@163.com

[1] School of Computer and Information Engineering, Henan University of Economics and Law, Zhengzhou 450000, China

Abstract

Cloud computing has evolved into an indispensable tool for facilitating scientific research due to its ability to efficiently distribute and process workloads in a virtual environment. Scientific tasks that involve complicated task dependencies and user-defined constraints related to quality of service (QoS) and time constraints require the efficient use of cloud resources. Planning these scientific workflow tasks represents an NP-complete problem, prompting researchers to explore various solutions, including conventional planners and evolutionary optimization algorithms. In this study, we present a novel, multistage algorithm specifically designed to schedule scientific workflows in cloud computing contexts. This approach addresses the challenges of efficiently mapping complex workflows onto distributed cloud resources while considering factors like resource heterogeneity, dynamic workloads, and stringent performance requirements. The algorithm uses the whale optimization algorithm (WOA) with a two-phase approach to shorten execution time, minimize financial costs, and effectively maintain load balancing.

Keywords: Cloud computing, Workflow scheduling, QoS, Whale optimization algorithm

Introduction

Cloud computing has emerged as a rapidly expanding area of distributed computing, providing scalable services over the Internet through hardware and software virtualization. This business model allows customers to access and purchase services under service-level agreements (SLAs) and pay for what they use, similar to conventional utilities [1]. The main strengths of cloud computing lie in its adaptability and flexibility, enabling individuals to access resources and services customized to their requirements remotely [2]. Cloud providers typically offer two resource provisioning plans for different user needs [3]. The first plan is the on-demand plan, where users request resources whenever needed, accommodating fluctuating and unpredictable inquiries. The second plan is the long-term preservation plan, where users reserve resources in advance, providing predictability and stability for their resource allocation. Leading cloud providers like Amazon EC2 and GoGrid offer services with both on-demand and reservation plans [4, 5].

Heterogeneous distributed computing environments consist of various computers, machines, and processors interconnected by high-speed networks [6]. These systems can collectively utilize their resource capacities to tackle complicated issues. Previously, scientific workloads were often executed on distributed grid computing, also known as e-science [7]. However, as cloud computing has evolved, it has been increasingly adopted for e-business purposes and for addressing e-science challenges. Cloud computing's widespread availability, cost-effectiveness, and flexibility, enabled by virtualization technology, have garnered significant interest from scientific and research communities. Scientific workflows, which are mathematical parallelizable processes, are implemented in various actual engineering tasks like FFT, GJ elimination, and LU decomposition. These workflows are commonly modeled as directed acyclic graphs (DAGs), with nodes reflecting application tasks and directed edges connecting data-dependent tasks [8]. The shapes and resource demands of such parallelizable applications vary widely. As users may not be able to expand existing infrastructure, distributed systems like cloud computing offer scalability, especially with the elasticity attribute, enabling the provision of services according to users' varying resource demands [9]. In a cloud environment, multiple virtual machines (VMs) can execute independent tasks simultaneously. Turnaround time represents one of the most critical QoS indicators, measuring the overall elapsed time between the initiation of the first task in an application and the completion of the last. A user's experience is directly affected by this time duration, known as makespan. Therefore, minimizing makespan is a significant objective function in this research [10].

The successful execution of scientific workflows relies on efficiently utilizing available resources. To achieve this, researchers focus on developing effective strategies to assign workflow tasks to computing resources, known as workflow scheduling. Workflow scheduling involves coordinating the execution of interdependent tasks, taking into account resource priority constraints [11]. Due to the NP completeness of this problem, researchers have focused on finding near-optimal solutions. To ensure workflows are well-defined and handled for future execution, an effective workflow management system (WMS) is essential. Figure 1 illustrates that a workflow scheduler (bridge) is required in a cloud environment to organize workflow tasks and allocate them to targeted resources. This workflow scheduler is crucial for efficient resource distribution and management of scientific workflows within cloud computing environments.

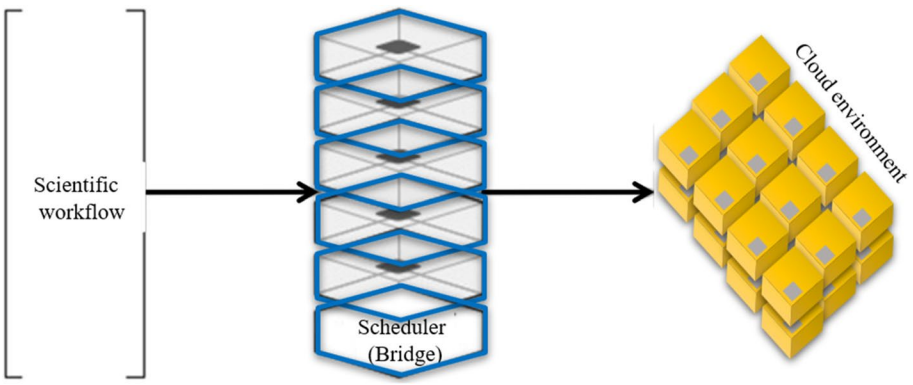

Fig. 1 Scientific workflow execution model in cloud computing

The task scheduling process presents a significant issue within cloud computing, especially for complex and diverse scientific workflows. Scientific workflows can be sensitive to large data volumes, complex processing, and multiple criteria simultaneously [12]. This complexity has motivated researchers to come up with strategies that optimize the handling of scientific workflows, with a particular focus on finding a balance between two conflicting QoS variables: cost and time [13]. QoS is a measure of user satisfaction with a given service and is often evaluated based on criteria like reliability, computational cost, and computational time. Balancing the conflicting goals of minimizing processing time and reducing costs can be challenging. Faster processing often requires more powerful and expensive resources, while cheaper resources may result in slower completion times [14]. To address this contradiction, it is necessary to shorten processing time while reducing costs, all while adhering to deadlines and budgets. This strategy seeks to maintain an ideal balance between meeting performance requirements and optimizing resource utilization.

This study aims to optimize scientific workflow scheduling performance within cloud computing environments. This is achieved by focusing on three key objectives: enhancing execution time, reducing financial costs, and maintaining effective load balancing across resources. By utilizing the whale optimization algorithm (WOA) in a multi-level approach, the proposed method seeks to minimize the makespan, ensuring that workflows are completed in the shortest possible time. Additionally, the algorithm aims to minimize the monetary costs associated with resource utilization by optimizing the task distribution among VMs. Finally, the study emphasizes maintaining load balancing, ensuring that the computational load is equally dispersed across all available resources to prevent bottlenecks and enhance overall system performance.

Related work

This section reviews significant contributions to scientific workflow scheduling in cloud-based systems, highlighting various methodologies and their effectiveness. Table 1 lays out a detailed comparison of these approaches, highlighting objectives, evaluation methods, and key findings.

Shi et al. [15] developed a resource allocation and task management framework in the cloud to handle scientific workflows in an elastic manner. This method is designed to efficiently execute critical workflow tasks within specified time and budget limitations. The approach encompasses four key phases: task preprocessing, task authorization, resource assignment on an elastic basis, and task scheduling. The performance assessment involves four types of real scientific workflow tasks with varying financial limitations. Additionally, uncertainties regarding task failures, processing delays, and estimated runtimes are considered in the assessment. The results demonstrate that in most scenarios, their mechanism outperforms other alternatives.

Aziza and Krichen [16] have introduced a novel model aimed at optimizing the execution time of interconnected tasks in the cloud while simultaneously reducing the workload and fulfilling strict demands and budget constraints. Their approach utilizes a hybrid process that incorporates a genetic algorithm to effectively model and improve cloud computing workflow scheduling. The heterogeneous earliest finish time (HEFT) generates the sample population. Through extensive experiments applied to actual

Table 1 Related work on workflow scheduling in cloud computing

Study	Methodology	Evaluation	Findings
[15]	Task scheduling and resource allocation framework; task preprocessing, task authorization, elastic resource assignment, and task scheduling phases	Performance was assessed with four types of real scientific workflow tasks under different financial constraints	Outperforms other alternatives in most scenarios; performance remains relatively unaffected by uncertainties in task runtime estimations, VM provisioning delays, and task failures
[16]	The hybrid model incorporates a genetic algorithm and HEFT heuristic for workflow scheduling	Extensive experiments on real-world scientific workflows, integrated GA-based module into WorkflowSim framework	Demonstrates superior performance compared to existing HEFT and other approaches, high efficiency in workflow scheduling
[17]	Genetic algorithms with innovative and modified genetic operators include a load-balancing routine	Comprehensive evaluation with an adaptive fitness function considering both cost and makespan, compared with state-of-the-art algorithms	Exhibits remarkable superiority over other approaches, achieving task scheduling with the lowest makespan and cost
[18]	A hybrid multi-objective optimization algorithm (HGSOA-GOA) combines the seagull optimization algorithm (SOA) and grasshopper optimization algorithm (GOA) and uses chaotic maps for random value generation	Evaluated via CloudSim and WorkflowSim tools, compared with SPEA2 algorithm using indicators like IGD and coverage rate	Outperforms other methods, evidenced by superior IGD, coverage rate, and other performance indicators
[19]	Task clustering and partial critical path algorithm incorporate dynamic voltage and frequency scaling (DVFS)	Simulations on various scientific applications (LIGO inspiral analysis, SIPHT, CyberShake, Montage)	Significant reductions in transmission costs and energy consumption, improved performance in scientific workflow execution
[20]	Two-stage approach with reliability-aware stepwise performance-to-power ratio (PPR) optimization method	Simulations in real-world and synthesized workflow applications	Achieves enhanced reliability while maintaining reduced energy constraints, outperforms competing workflow mapping methods in terms of both reliability and energy efficiency

scientific tasks, they have demonstrated the superior performance of their developed approach compared to existing HEFT and other approaches analyzed in their research. The experiments clearly demonstrate the high efficiency of their approach, making it a viable approach to cloud workflow planning. To implement their proposed model effectively, they have developed a GA-based module, which has been seamlessly adapted to the CloudSim-based WorkflowSim framework.

Workflow optimization heavily relies on efficient task scheduling, considered a widely recognized NP-hard issue. The genetic algorithm has been used in numerous strategies for cloud task scheduling, but the strategy puts forth by Iranmanesh and Naji [17] stands out as more efficient than others. This is due to the application of altered genetic operators and the integration of a load-balancing mechanism. Notably, their method employs a heuristic solution to generate one primary population chromosome and utilizes a heuristic method to generate the remaining primary population chromosomes. To ensure a comprehensive evaluation, an adaptive fitness function is employed, taking both cost and makespan into consideration. By introducing a load-balancing routine, their algorithm maximizes resource efficiency during execution. The results are evaluated against leading algorithms in this domain to gauge the performance of their algorithm. The findings reveal that the approach exhibits substantial advantages over other approaches, achieving the most efficient and cost-effective task scheduling.

Mohammadzadeh and Masdari [18] have introduced a mixed multi-objective optimization algorithm named HGSOA-GOA, combining the grasshopper optimization algorithm (GOA) and seagull optimization algorithm (SOA). The algorithm utilizes chaotic patterns to generate stochastic values, achieving a balance between exploiting and exploring, resulting in improved convergence rates. The HGSOA-GOA algorithm addresses scheduling challenges for scientific workflows in cloud settings, incorporating multiple factors like throughput, energy, cost, and makespan. It aims to optimize these objectives simultaneously, enabling better decision-making for task assignments in a multi-cloud scenario. In this approach, Pareto Solutions are chosen using the knee-point strategy, which helps identify an optimal compromise solution. This selected solution is then used to assign scientific workflow tasks in multi-cloud environments. To evaluate the performance of HGSOA-GOA, comprehensive analyses are performed via the CloudSim and WorkflowSim simulators, and the outcomes are evaluated against those obtained from the SPEA2 model. The findings demonstrate that the HGSOA-GOA algorithm outperforms alternative methods, as evidenced by indicators including IGD (inverted generational distance), coverage rate, and others.

Choudhary et al. [19] have implemented a task clustering and partial critical path algorithm to enhance the efficiency of scientific workflow execution. This algorithm groups fine-grained tasks into jobs and assigns sub-deadlines recursively to tasks situated on the partial critical path. This approach helps optimize the workflow and improves the overall performance. Additionally, they address the energy efficiency aspect by incorporating the dynamic voltage and frequency scaling (DVFS) method. This procedure constantly adjusts the voltage and frequency of computing nodes' processors based on workload, allowing for energy-saving opportunities during execution. To validate their proposed framework, simulations are performed on various scientific applications, including LIGO inspiral analysis, SIPHT, CyberShake, and Montage. The results from these

simulations demonstrate that the implemented task clustering and DVFS techniques effectively address the mentioned issues. The analysis of the results reveals considerable savings in transmission costs and energy utilization, indicating the efficiency of the suggested approach in achieving energy savings and improved performance in scientific workflow execution.

Khaleel [20] addressed the scientific workflow scheduling problem as a multi-objective optimization challenge, aiming to find an equilibrium between scheduling reliability and energy efficiency. They suggest a two-step approach to reach this compromise. In the first step, tasks requiring moderate computational demands are placed on fog resources, while tasks with higher computational demands are assigned to cloud resources. This strategy avoids high-failure-rate resources, thereby enhancing scheduling reliability. In the second step, they employ the reliability-sensitive stepwise performance-to-power ratio (PPR) optimization method to minimize energy usage significantly. This involves measuring the machine's utilization rates using PPR. The PPR is determined by the ratio of transactions completed at certain times to the energy consumed. Through simulations in real-world and synthesized workflow applications, the proposed approach, called dis-RMEE, achieves significantly enhanced reliability while maintaining reduced energy constraints. It outperforms its competing workflow mapping methods in terms of both reliability and energy efficiency.

Methods

This section introduces a many-objective algorithm for task scheduling. It consists of two main components: preprocessing and a scheduling approach based on the WOA. Before the scheduling process begins, the workflow and its associated tasks are preprocessed to gather essential information. This step involves analyzing the workflow's structure, identifying task dependencies, and assessing the resource requirements of each task. The algorithm searches for optimal solutions by iteratively updating the positions of candidate solutions, known as whales, in the search space. In the context of task scheduling, the WOA algorithm is adapted to find an optimal assignment of tasks to available computing resources. The positions of the whales represent different possible task-resource assignments, and the algorithm iteratively explores and updates these assignments to improve the overall performance of the workflow. By employing the WOA-based scheduling approach, our multi-objective hybrid algorithm aims to optimize multiple objectives, such as minimizing makespan (processing time), reducing resource utilization costs, and meeting deadline constraints.

Problem statement

In the context of workflow scheduling in cloud computing, workflows are expressed as DAGs, denoted by $G=(V, E)$, where V corresponds to the set of points that signify individual workflow tasks and E indicates the set of edges representing task dependencies. In a DAG, each task has a specific execution order, and a parent task must be completed before any of its child tasks can start execution. Each workflow task in the DAG has various parameters that are crucial for scheduling and resource allocation. Execution time denotes the time required for a task to be executed on a particular resource. Different tasks may have different execution times depending on their computational complexity. Dependencies

stand for the relationships between tasks, where a child's task depends on the completion of its parent task(s). These dependencies need to be considered during scheduling to ensure correct execution order.

Workflow tasks can vary in their computational and data intensity. Some tasks may be computation intensive, involving complex computations, while others may be data intensive, requiring significant data handling and transfer. In some cases, tasks may be both data and computation intensive, making the scheduling process more challenging. The cloud structure is formed of multiple data centers, each containing a number of physical machines. These machines are equipped with data storage resources and bandwidth capabilities. In cloud computing, resources are depicted as VMs. VMs are virtualized instances that can be deployed and managed on the physical machines within data centers. Each VM has fixed attributes such as bandwidth, processing capabilities (measured by the number of processing elements or PEs), and storage cost per unit of time. The processing capacity of VMs is calculated by Eq. 1.

$$C_i = (MIPS_i \times PE) \tag{1}$$

The capacity of n VMs is calculated by Eq. 2.

$$C = \sum_{i=1}^{n} C_i \tag{2}$$

The VMs' loads are computed by Eq. 3 as the percentage of the total number of tasks performed by VMs to their capacities.

$$L_{vm_i} = \frac{TL}{C_i} \tag{3}$$

Equation 4 calculates the load of all VMs in the cloud system. The time taken to finish the i^{th} task is the sum of the duration needed to retrieve the necessary data for execution (time to obtain data for task i) and the execution time of the task itself. This time of completion is a vital measurement to consider in task scheduling and load balancing algorithms, as it directly impacts the overall makespan of the workflow and the performance of the system. Equation 5 calculates the completion time of a workflow task i.

$$L = \sum_{i=1}^{n} L_{vm_i} \tag{4}$$

$$Time(t_i) = Time_E((t_i, VM_k) + Time(Trans_{t_i,t_j}) \tag{5}$$

Equation 5 defines $Trans_{t_i,t_j}$ as the duration of data transmission between tasks i and j, while $Time_E(t_i, VM_k)$ represents the execution time of the i^{th} task on VM_k. The transmission duration is estimated using Eq. 6, while Eq. 7 provides the ability to compute the execution time.

$$Trans(t_i, t_j) = \frac{sizeof(t_i, t_j)}{\beta(VM_k, VM_m)} \tag{6}$$

$$T_E = \frac{l_i}{C_{m_j}} \tag{7}$$

Within the aforementioned equations, the term $sizeof(t_i, t_j)$ corresponds to the data size transmitted between tasks i and j, and $\beta(VM_k, VM_m)$ denotes the bandwidth of data centers in which VM_k and VM_m are situated. When both VMs are located within the same data center, the transmission cost is effectively reduced to zero. Equation 7 incorporates the variables l_i and C_{m_j}, where l_i represents the length of task i and C_{m_j} denotes the processing capacity of VM_j, as determined using Eq. 2. The makespan corresponds to the last task's completion time within the workflow, calculated by Eq. 8.

$$Makespan = FT_{i=1}^{n}[task_i time] \tag{8}$$

Equation 8 defines the makespan as the task's finish time (FT). The monetary cost (MC) associated with a workflow encompasses both the data transfer and execution costs between workflow tasks. This cost can be computed using Eq. 9. The MC further comprises the total transfer cost (TTC) determined using Eq. 10 and the total execution cost (TEC) evaluated with Eq. 10.

$$MC = TTC + TEC \tag{9}$$

$$TTC_{wi} = \sum_{i=1}^{n} \frac{size(t_i, t_j)}{\beta cost} \tag{10}$$

$$TEC_{wi} = \sum_{i=1}^{n} vm_i^{time} vm_i^{cost} \tag{11}$$

Equation 10 presents the relationship between the size, representing the data size and cost, which denotes the bandwidth cost for data transfer. Additionally, load balance is expressed as the variance of the load across all nodes, as depicted in Eq. 12. Smaller values of load balance indicate more effective load management.

$$L = \sqrt{\frac{\sum_{i=1}^{m} (L_{vm_i} - R)^2}{m}} \tag{12}$$

In Eq. 12, L_{vm_i} denotes the load of vm_i, R represents the mean load across all VMs, and m indicates the total number of VMs in the system.

Whale optimization algorithm

The WOA is a computational intelligence algorithm that draws inspiration from the foraging behavior of humpback whales, specifically their bubble-net hunting mechanism. Humpback whales are highly intelligent creatures with brain cells called spindle cells, similar to humans [21]. They exhibit complex behaviors, including communication through their own dialect, making them an intriguing inspiration for optimization algorithms. The WOA algorithm begins by randomly generating a population of candidate solutions across the search space. Each candidate solution's fitness is calculated based on the objective function, and

the best solution is identified. The WOA operates in three main phases: encircling prey, bubble-net hunting, and prey searching.

Once the humpback whales find the best search agent (the one with the highest fitness value), the rest of the whale population starts encircling that agent. This behavior is a part of the WOA's exploration phase. Search agents update their positions during this process as they approach prey (the best search agent). The mathematical model for the current position update during the encircling behavior in the WOA can be described as follows:

$$WX_{curr}(t+1) = WX_{best}(t) - A \times D_{dis} \tag{13}$$

$$A = 2a \times r_1 - a \tag{14}$$

$$D_{dis} = |C \times WX_{best}(t) - WX_{curr}(t)| \tag{15}$$

$$C = 2 \times r_2 \tag{16}$$

D_{dis} represents the distance between the current solution $WX_{curr}(t)$ and the best solution $WX_{best}(t)$ at iteration t. The values of r_1 and r_2 are random numbers within the range [0, 1]. The variable a is the shrinking decreased linearity value, which starts at two and gradually decreases to 0 as the iterations progress. If the WOA algorithm chooses the first method, i.e., the shrinking encircling mechanism, for updating the solution, it follows the same process as described earlier using the decreasing value of Eq. 18. This method involves calculating the encircling vector and updating the current solution's position based on the shrinking decreased linearity value and the current iteration number. On the other hand, if the WOA algorithm opts for the second method, i.e., the spiral update based on random selection probability, the population of whales will navigate around the current optimal solution denoted as WX_{best}. The process for this method is defined as follows:

$$WX_{curr}(t+1) = Dt'_{dis} \times e^{bk} \times \cos(2\pi k) + WX_{best}(t) \tag{17}$$

$$D'_{dis} = |WX_{best}(t) - WX_{curr}(t)| \tag{18}$$

k varies randomly between $[-1, 1]$, and b controls the magnitude of the spiral movement. The algorithm switches between the shrinking and spiral paths for solution updates based on a probability distribution. The equation to determine whether to use the shrinking encircling mechanism or the spiral update method is as follows:

$$WX_{curr}(t+1) = \begin{cases} WX_{best}(t) - A \times D_{dis} if r_3 \geq 0.5 \\ D'_{dis} \times e^{bk} \times \cos(2\pi k) + WX_{best}(t) if r_3 < 05 \end{cases} \tag{19}$$

The whales engage in a random search for their optimal prey, denoted as WX_{best}, by exploring a random position (WX_{rand}) instead of relying on the best solution (also WX_{best}), which is determined using the following equations.

$$WX_{curr}(t+1) = WX_{rand}(t) - A \times D_{dis} \tag{20}$$

$$D_{dis} = |C.WX_{rand}(t) - WX_{curr}(t)| \tag{21}$$

The WOA algorithm utilizes four parameters, namely a, A, C, and $r_{3,}$ to update solutions. If r_3 is greater than 0.5, Eq. 17 is employed for solution updating. On the other hand, if r_3 is less than 0.5, Eqs. 20 and 21 are used. Furthermore, depending on the value of $|A|$, the solution can also be updated using Eqs. 13 and 15. The WOA updates the position vector of the solution until the specified conditions are satisfied. Upon meeting the criteria for termination, WOA's best solution is returned as WX_{best}. Figures 2 and 3 depict how solutions are updated in the WOA, demonstrating that solutions have the flexibility to move close to a solution in random positions or follow a spiral path.

Proposed algorithm

The proposed algorithm utilizes WOA to optimize both execution time (makespan) and monetary cost of workflows while ensuring a balanced load distribution across all nodes. WOA stands out among other meta-heuristics due to its speed and faster convergence, making it a favorable choice for this application. The algorithm finds its application in various cloud computing areas, including VM scheduling and placement, among others. To effectively execute the algorithm, the scheduler is assumed to possess knowledge of task dependencies within the workflow, and the execution times of individual tasks are predetermined. The main objective of the suggested algorithm is to efficiently allocate cloud resources to workflow tasks, thereby minimizing both execution time and monetary cost. Achieving this allocation is crucial for ensuring the effective utilization of cloud resources. In addition to this, the scheduler needs to take other relevant parameters into account while scheduling cloud resources, further enhancing the overall efficiency and effectiveness of the scheduling process.

Before applying the WOA, the proposed algorithm employs a preprocessing step to organize resources and tasks for optimization. Initially, the algorithm prioritizes tasks based on the number of descendants they have, giving priority to tasks with a large number of

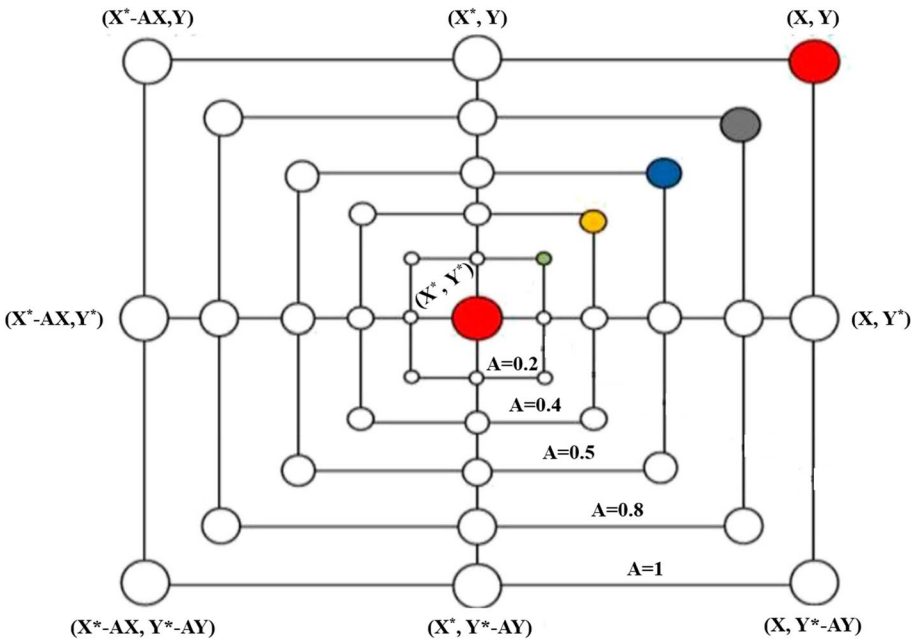

Fig. 2 Bubble-net search strategy

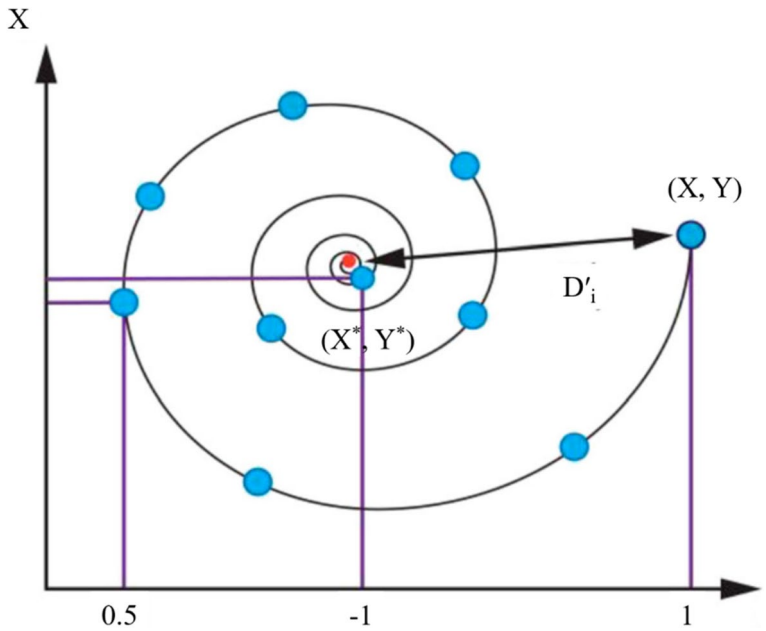

Fig. 3 Position update in a spiral

descendants. These tasks tend to become bottlenecks for cloud resources and consequently lead to longer execution times. Similarly, the algorithm also sorts cloud resources according to their processing power, distinguishing between low-processing power and high-processing power resources. It maintains two lists of resources for processing workflow tasks. Parent tasks, which require significant processing time due to their dependencies, are handled by high-processing nodes to expedite dependency elimination. Once the parent tasks are completed, the children's tasks are addressed based on their place in the workflow graph. Leaf tasks are performed using low-processing nodes, while parent and intermediate tasks are assigned to available high-processing nodes. To ensure load balance across resources, the algorithm periodically switches tasks between two lists: the dependency list containing tasks with dependencies and the independent list containing tasks without dependencies. The WOA is then utilized for resource-to-task assignments for both lists. Each task begins execution with random whale positions and velocities, and each whale's performance is assessed using the fitness function. Variables are adjusted iteratively through multiple iterations until a stopping criterion is reached. The fitness function, used to calculate the fitness of a whale, considers both the execution time and the cost, as represented in Eq. 22. Figure 4 depicts the flowchart of the proposed algorithm.

$$F = w\frac{max^{ms} - ms}{max^{ms} min^{ms}} + w\frac{max^{mc} - mc}{max^{mc} - min^{mc}} + w\frac{max^{lb} - lb}{max^{lb} - min^{lb}} \tag{22}$$

Results and discussion

In this section, we provide an overview of performance measures, simulation settings, and the workflow design employed in the proposed algorithm. A collection of interconnected computational tasks characterizes a workflow. Various scientific workflows

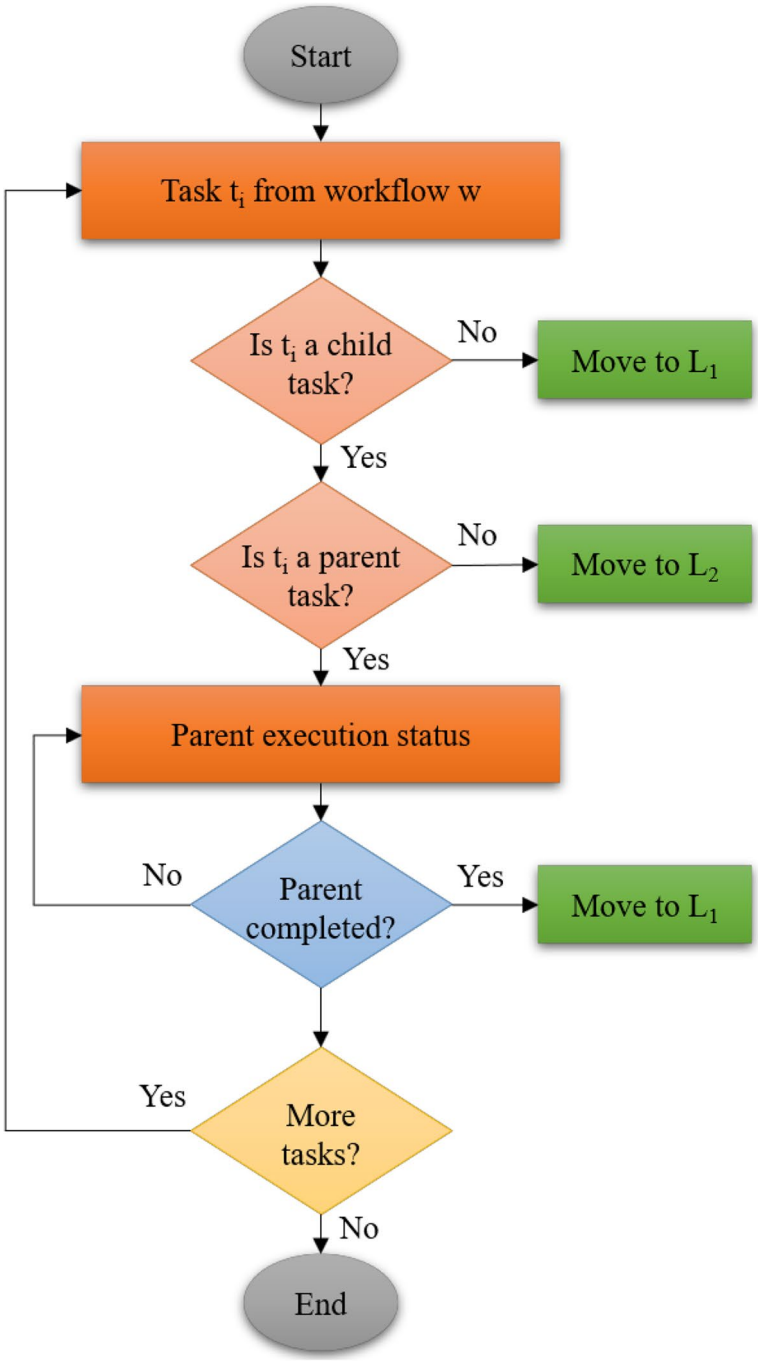

Fig. 4 The preprocessing step in the proposed algorithm

like Pan-STARRS, Epigenomics, Cybershake, Montage, and LIGO have been documented in the literature. For this paper, the Pan-STARRS scientific workflow is adopted as the framework for task execution. The Pan-STARRS project is dedicated to ongoing sky monitoring to identify movable or varying celestial objects. The PS1 telescopic instrument is employed for this purpose. The astronomy data generated by this project is managed by John Hopkins University and Microsoft, utilizing two

distinct workflows: PSMerge and PSLoad. The PSMerge workflow focuses on database updates, while the PSLoad workflow collects data from the telescope and records them in the database. Detailed depictions of these workflows are illustrated in Figs. 5 and 6. Comprehensive characteristics of these workflows are outlined in Table 2.

The simulation of the cloud environment and validation of the proposed scheduling model are carried out using the CloudSim simulator. Two distinct host types were deployed: HP ProLiant ML110 G4 and HP ProLiant ML110 G5. These host types exhibit variations in energy consumption rates, quantified in watts per second (Ws-1). Energy consumption rates for deployed hosts are specified. Also, a separate value is provided for the energy consumption associated with transferring 1 gigabyte (GB) of data. Four unique VM configurations were implemented in the simulation environment. These machines differ in terms of their processing power, measured in millions of instructions per second (MIPS), and their available random access memory (RAM) capacity, specified in megabytes (MB). Each VM configuration is broken down:

- VM type 1: This VM offers 500 MIPS of processing power and is equipped with 613 MB of RAM.
- VM type 2: This VM configuration provides 1000 MIPS of processing power and boasts 1740 MB of RAM.
- VM type 3: The third VM configuration delivers 2000 MIPS of processing power while maintaining the same 1740 MB of RAM capacity as VM type 2.
- VM type 4: This specialized VM configuration is designated for executing the scientific workflow. It offers 2500 MIPS of processing power but has a reduced RAM capacity of 870 MB.

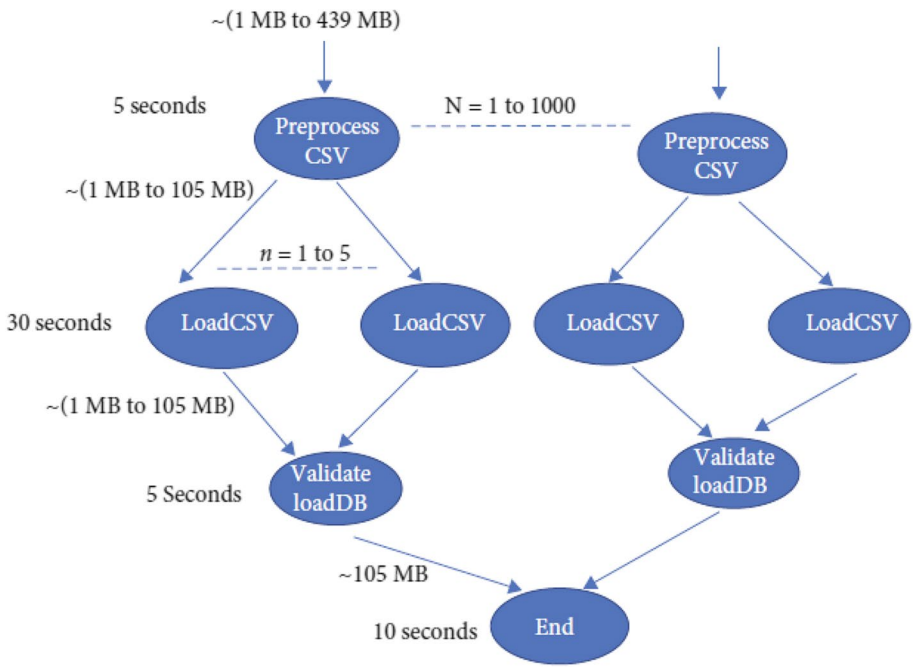

Fig. 5 PSLoad workflow

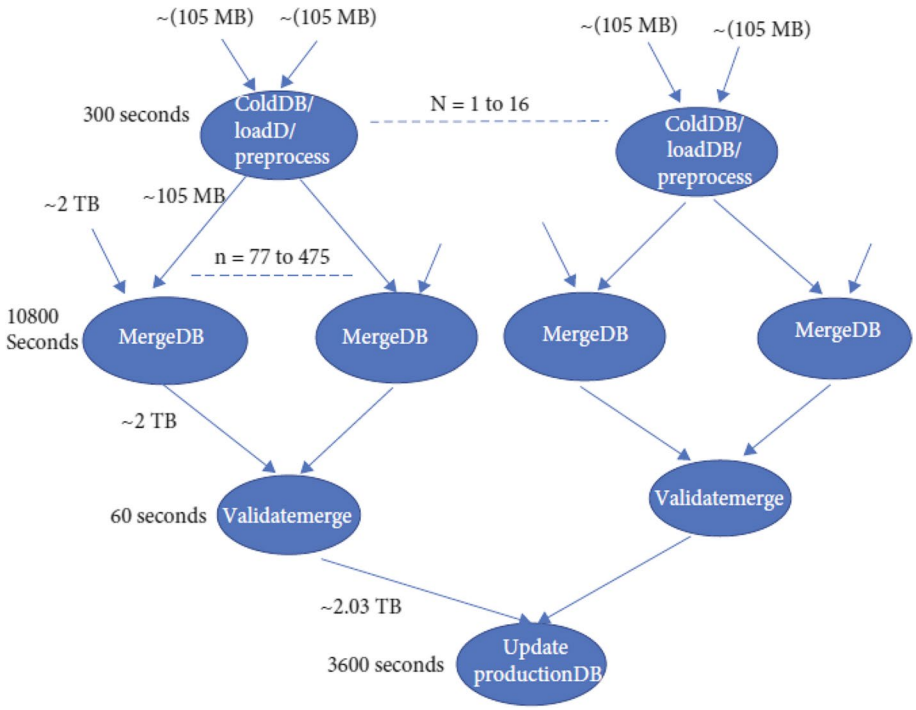

Fig. 6 PSMerge workflow

The average VM startup time was established within the simulation to align with scientific workflow requirements. Furthermore, the average bandwidth between VMs was configured to approximate the bandwidth capabilities offered by a prominent cloud computing service provider, Amazon Web Services (AWS). A real-world scientific workflow, derived from the Pan-STARRS astronomical survey project, was chosen as the benchmark for this simulation. This workflow is categorized into three distinct groups based on the number of tasks it entails. A detailed breakdown of these task groupings is provided in Table 2.

The assessment of the proposed method is carried out through a comparative analysis against PESVMC [22] and EVMP [23], considering key performance metrics encompassing execution time (makespan), energy consumption, and resource utilization. The makespan, representing the time taken for the complete execution of the scientific workflow from its initial tasks to the final task, is evaluated using Eq. 23. Here, $subTime_{workflow}$ denotes the submission time of the workflow. Energy consumption quantifies the overall energy expended by the servers during the execution of the scientific workflow. This is computed utilizing Eq. 24, where ecr_{ijk} signifies the energy consumption rate of VM j on host k and x_{ijk} indicates the mapping of task i onto VM j at host k. The variable x_{ijk} is set to 1 if task i is scheduled on VM j at host k for execution; otherwise, it is 0. The average resource utilization denotes the proportion of allocated computing resources for accomplishing the tasks of the scientific workflow relative to the total computing resources available on the server. This is calculated using Eq. 25.

$$makespan = \max\left(ft_{ijk}\right) - subTime_{workflow} \tag{23}$$

Table 2 Workflow description

Workflows	Task count	Child task count	Edge count	Tasks types	Task count	Input file size (MB)	Deadline (second)	Tasks length
PSMerge_small	80	79	234	UpdateProductionDB	1	2,232,008.604	3600	1,800,000–9,000,000
				ValidateMerge	1	2,199,023.256	60	30,000–150,000
				MergeDB	70	540,000–27,000,000	10,800	540,000–27,000,000
				ColdDB/LoadDB/Preprocess	1	104.86 and 104.86	300	150,000–750,000
PSMerge_medium	841	836	2505	UpdateProductionDB	1	2,232,008.604	3600	1,800,000–9,000,000
				ValidateMerge	5	2,199,023.256	6	30,000–150,000
				MergeDB	830	104.86 and 2,199,023.256	10,800	540,000–27,000,000
				ColdDB/LoadDB/Preprocess	5	104.86 and 104.86	300	150,000–750,000
PSMerge_large	7622	7606	22,815	UpdateProductionDB	1	2,232,008.604	3600	1,800,000–9,000,000
				ValidateMerge	16	2,199,023.256	60	30,000–150,000
				MergeDB	7589	104.86 and 2,199,023.256	10,800	540,000–27,000,000
				ColdDB/LoadDB/Preprocess	16	104.86 and 104.86	300	150,000–750,000
PSLoad_small	4	3	4	ValidateLoadDB	1	97.52	5	2500–12,500
				LoadCSV	1	97.52	30	15,000–75,000
				PreprocessorCSV	1	97.52	5	2500–12,500
				End	1	104.86	10	5000–25,000
PSLoad_medium	489	389	776	ValidateLoadDB	100	1.05–104.86	5	2500–12,500
				LoadCSV	288	1.05–104.86	30	15,000–75,000
				PreprocessorCSV	100	6.29–362.81	5	2500–12,500
				End	1	104.86	10	5000–25,000

Table 2 (continued)

Workflows	Task count	Child task count	Edge count	Tasks types	Task count	Input file size (MB)	Deadline (second)	Tasks length
PSLoad_ large	5084	4084	8166	Validate-LoadDB	1000	1.05–104.86	5	2500–12,500
				LoadCSV	3083	1.05–104.86	30	15,000–75,000
				Preproc-essorCSV	1000	1.05–438.3	5	2500–12,500
				End	1	104.86	10	5000–25,000

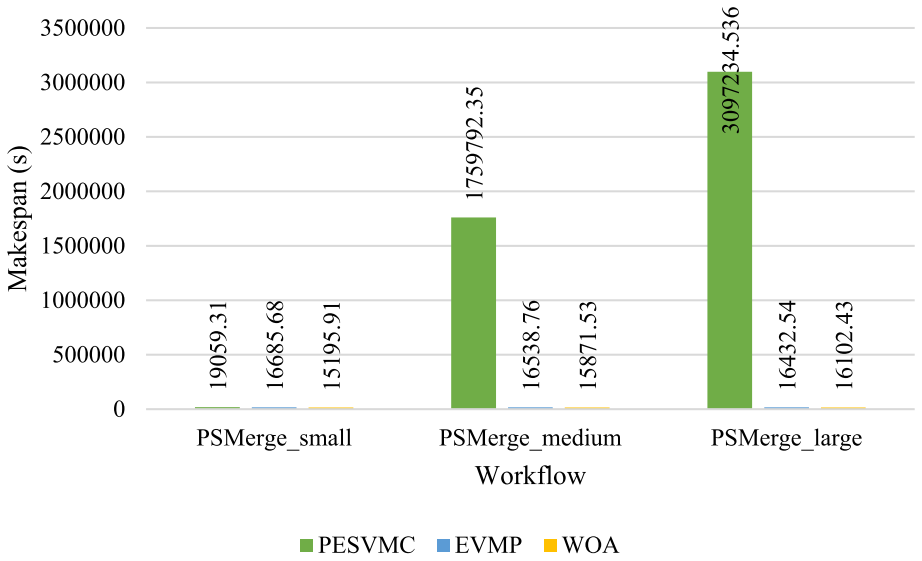

Fig. 7 Makespan comparison for PSMerge workflow

$$EC = \sum_{k=1}^{|Host_a|} \sum_{j=1}^{|VM_k|} \sum_{i=1}^{T} x_{ijk}.ecr_{ijk}.et_{ijk} \tag{24}$$

$$ARU = \sum_{k=1}^{|Host_a|} \sum_{j=1}^{|VM_k|} \sum_{i=1}^{T} l_{ijk}.x_{ijk} \div \sum_{k=1}^{|Host_a|} c_k.at_k \tag{25}$$

The comparison of makespan across different methods, including WOA, PESVMC, and EVMP, is presented for both PSMerge and PSLoad workflows, each with varying numbers of tasks. Figures 7 and 8 depict simulation outcomes concerning makespan. Results show that the WOA algorithm is superior to the others in terms of makespan efficiency in both workflows. On average, there is a reduction of 10% and 98% in makespan when compared to the EVMP and PESVMC algorithms, respectively. The inferior efficiency of the current algorithm results from its failure to consider

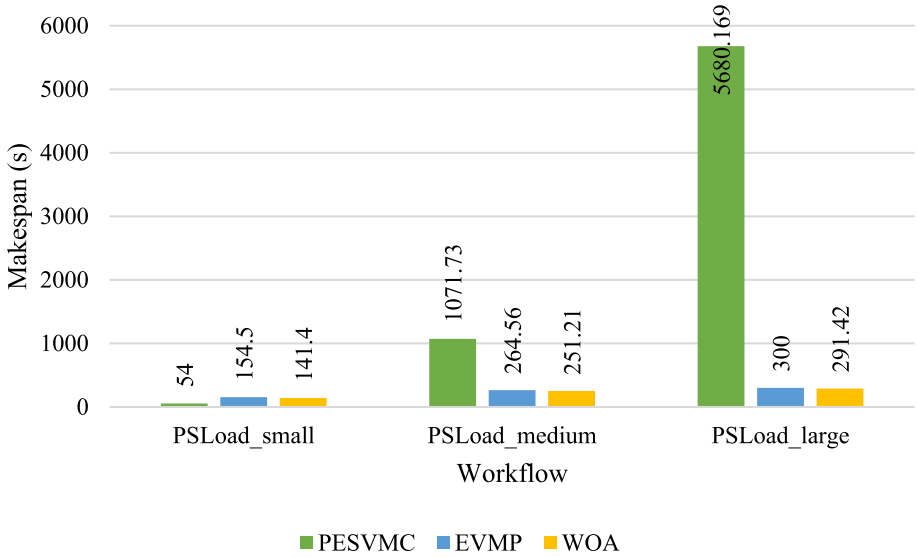

Fig. 8 Makespan comparison for PSLoad workflow

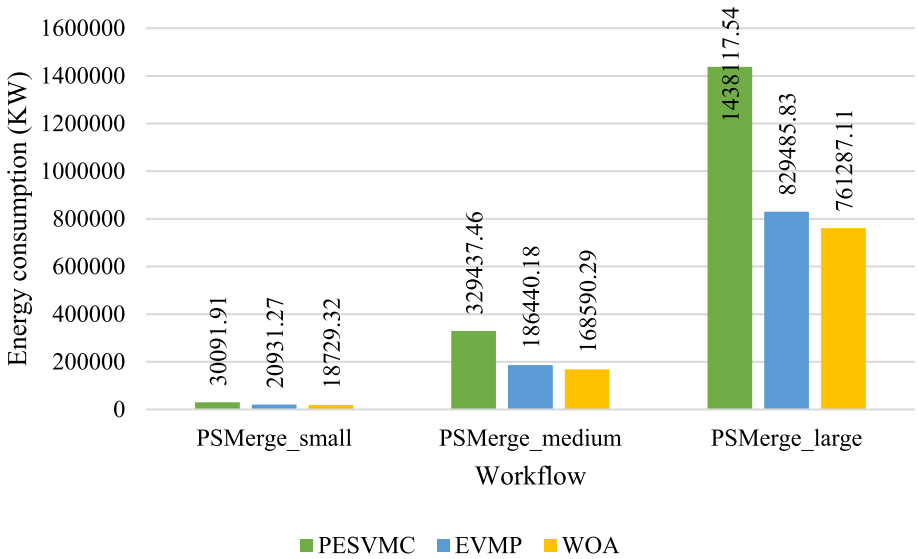

Fig. 9 Energy consumption comparison for PSMerge workflow

child-parent relationships during task scheduling on VMs. This deficiency has had a noticeable impact on the workflow's makespan.

The comparison of total energy consumption across different methods, including WOA, PESVMC, and EVMP, is conducted for both PSMerge and PSLoad scientific workflows with varying task counts. The test results concerning total energy consumption, expressed in kilowatts (kWs), are illustrated in Figs. 9 and 10. The results suggest that the WOA algorithm demonstrates an impressive reduction in energy usage. This achievement stems from the WOA algorithm's ability to allocate resources in alignment with workflow task requirements, effectively curbing energy usage. Furthermore, the suggested algorithm takes into account child-parent relationships in task scheduling,

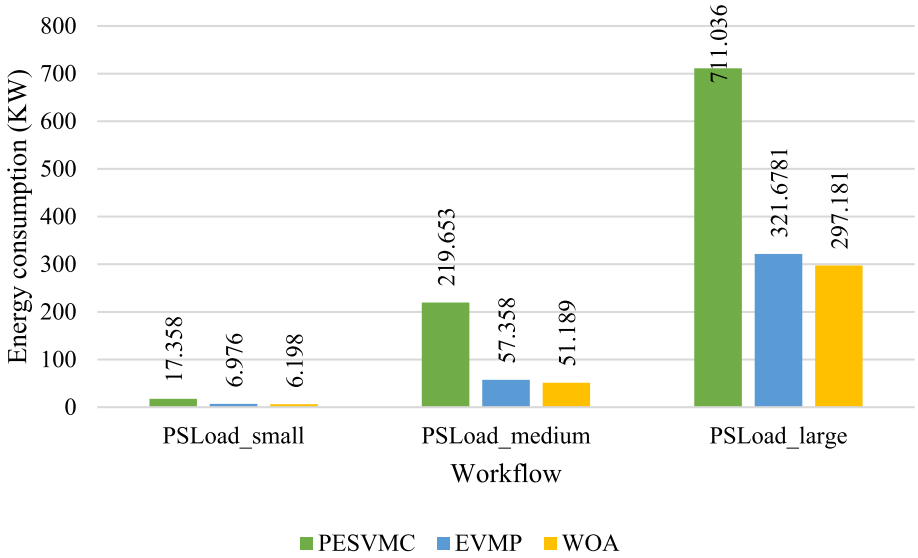

Fig. 10 Energy consumption comparison for PSLoad workflow

leading to a reduction in data transfer-related energy consumption. On average, the WOA algorithm showcases a remarkable reduction of 20% and 42% in energy consumption when compared to the EVMP and PESVMC algorithms, respectively. These substantial energy savings highlight the potential of the WOA algorithm in optimizing resource utilization and energy efficiency in the context of workflow scheduling.

The assessment of average resource utilization across different methods, including WOA, PESVMC, and EVMP, is conducted for both PSMerge and PSLoad scientific workflows, each encompassing varying numbers of tasks. The experimental findings regarding average resource utilization are graphically presented in Figs. 11 and 12. The findings underscore the better resource utilization performance of the WOA algorithm when compared to EVMP and PESVMC. The dynamic nature of the WOA algorithm contributes to its enhanced performance. The proposed algorithm strategically creates new VMs when the currently deployed VMs prove inadequate to meet task deadlines. This dynamic provisioning ensures optimal resource utilization. Additionally, the utilization of a VM migration policy facilitates resource consolidation, leading to impressive gains in resource utilization. On average, the WOA algorithm achieves a noteworthy increase of 10% and 8.6% in resource utilization over the baseline algorithms. These substantial improvements validate the effectiveness of the WOA algorithm in dynamically adapting to resource demands and efficiently utilizing available computing resources.

Conclusion

In the ever-evolving environment of cloud computing, where the submission of scientific workflows by user groups is a prevalent scenario, the efficient scheduling of extensive task sets within these workflows has emerged as a pivotal challenge. Task scheduling optimization holds paramount significance in cloud computing, aiming to minimize workflow execution time, optimize cloud resource consumption, and reduce execution costs for users. To cope with these imperatives, this study introduced a novel

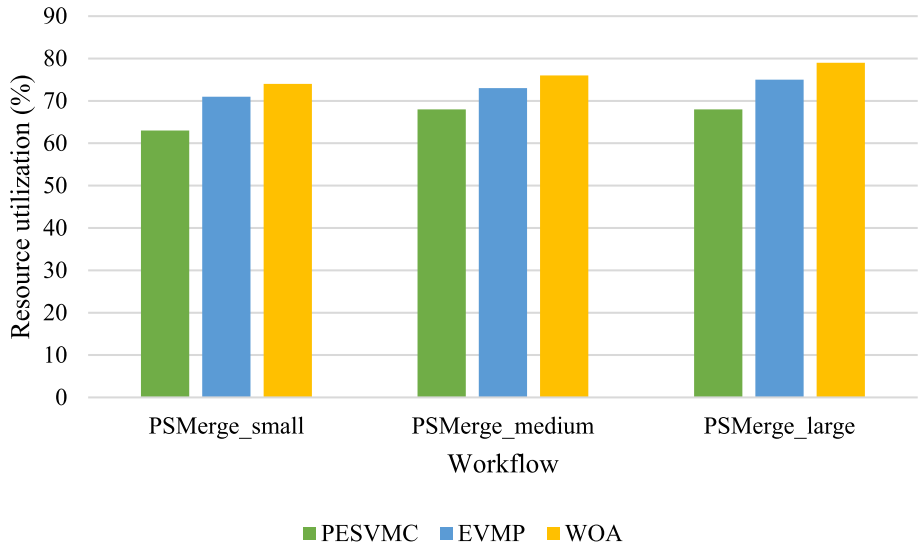

Fig. 11 Resource utilization comparison for PSMerge workflow

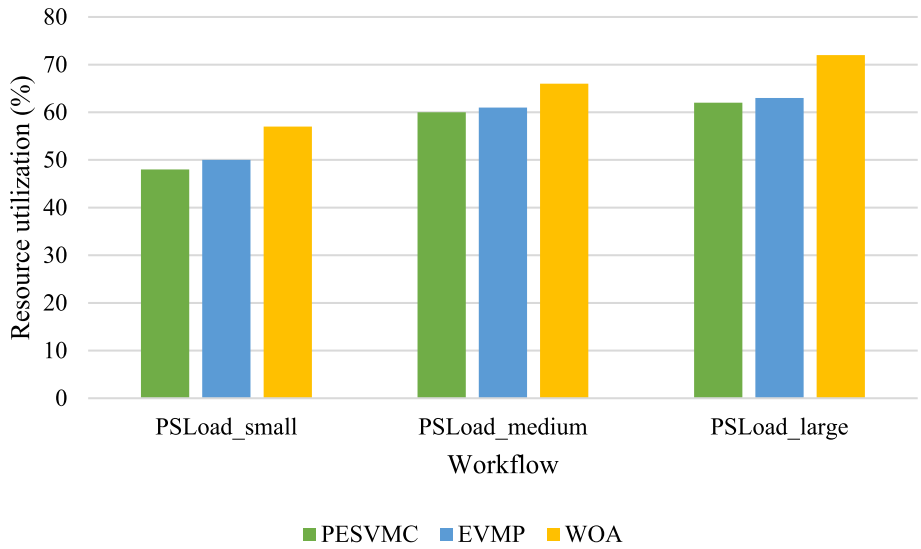

Fig. 12 Resource utilization comparison for PSLoad workflow

WOA-based task scheduling technique. This algorithm achieved an optimal distribution of tasks across available computing resources. The algorithm capitalizes on the concept of "whales," which symbolize potential task-resource assignments. Through an iterative process, the algorithm explores and refines these assignments to enhance the overall workflow performance, thereby striving to cut down execution time, resource usage, and associated budgets. The findings of comprehensive simulations provide compelling evidence for the better efficiency of the proposed WOA-based algorithm. Notably, in comparison to previous algorithms, the proposed approach excels in energy consumption, resource utilization, and makespan.

Abbreviations

AWS	Amazon Web Services
DAG	Directed acyclic graphs
DVFS	Dynamic voltage and frequency scaling
FT	Finish time
GOA	Grasshopper optimization algorithm
HEFT	Heterogeneous earliest finish time
IGD	Inverted generational distance
MC	Monetary cost
PPR	Performance-to-power ratio
QoS	Quality of service
SOA	Seagull optimization algorithm
SLA	Service-level agreement
TEC	Total execution cost
TTC	Total transfer cost
VM	Virtual machine
WOA	Whale optimization algorithm
WMS	Workflow management system

Acknowledgements
Not applicable.

Author's contributions
XZ contributed to writing—original draft preparation and conceptualization.

Funding
No funding.

Availability of data and materials
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The author declares no competing interests.

Received: 15 July 2024 Accepted: 11 August 2024
Published online: 19 August 2024

References

1. Hayyolalam V, Pourghebleh B, Kazem AAP, Ghaffari A (2019) Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques. Int J Adv Manufact Technol 105(1–4):471–498
2. Wang X, Sun Y, Sun Q, Lin W, Wang JZ, Li W (2023) HCIndex: a Hilbert-curve-based clustering index for efficient multi-dimensional queries for cloud storage systems. Clust Comput 26(3):2011–2025
3. Hayyolalam V, Pourghebleh B, Chehrehzad MR, Pourhaji Kazem AA (2022) Single-objective service composition methods in cloud manufacturing systems: recent techniques, classification, and future trends. Concurr Comput 34(5):e6698
4. Yakubu IZ, Murali M (2023) An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment. J Ambient Intell Humaniz Comput 14(3):2981–2992
5. Sefati S, Mousavinasab M, Zareh Farkhady R (2022) Load balancing in cloud computing environment using the grey wolf optimization algorithm based on the reliability: performance evaluation. J Supercomputing 78(1):18–42
6. Al-Jumaili AHA, Muniyandi RC, Hasan MK, Paw JKS, Singh MJ (2023) Big data analytics using cloud computing based frameworks for power management systems: status, constraints, and future recommendations. Sensors 23(6):2952
7. He J (2022) Cloud computing load balancing mechanism taking into account load balancing ant colony optimization algorithm. Comput Intell Neurosci 2022:3120883
8. Mangalampalli S et al (2023) Prioritized task-scheduling algorithm in cloud computing using cat swarm optimization. Sensors 23(13):6155
9. Praveenchandar J, Tamilarasi A (2021) Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing. J Ambient Intell Humaniz Comput 12(3):4147–4159
10. Dubey K, Sharma SC (2021) A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing. Sustain Comput 32:100605
11. M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, (2020) Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. J Grid Comput 18:1–30
12. Kamanga CT, Bugingo E, Badibanga SN, Mukendi EM (2023) A multi-criteria decision making heuristic for workflow scheduling in cloud computing environment. J Supercomput 79(1):243–264
13. Mikram H, El Kafhali S, Saadi Y (2024) HEPGA: a new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment. Simul Model Pract Theory 130:102864

14. Asghari Alaie Y, Hosseini Shirvani M, Rahmani AM (2023) A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach. J Supercomputing 79(2):1451–1503
15. Shi J, Luo J, Dong F, Zhang J, Zhang J (2016) Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints. Clust Comput 19:167–182
16. Aziza H, Krichen S (2020) A hybrid genetic algorithm for scientific workflow scheduling in cloud environment. Neural Comput Appl 32:15263–15278
17. Iranmanesh A, Naji HR (2021) DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing. Clust Comput 24:667–681
18. A Mohammadzadeh, M Masdari (2021) Scientific workflow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm. J Ambient Intellig Humanized Comput 14:3509–3529
19. Choudhary A, Govil MC, Singh G, Awasthi LK, Pilli ES (2022) Energy-aware scientific workflow scheduling in cloud environment. Clust Comput 25(6):3845–3874
20. Khaleel MI (2022) Multi-objective optimization for scientific workflow scheduling based on performance-to-power ratio in fog–cloud environments. Simul Model Pract Theory 119:102589
21. Al-Moalmi A, Luo J, Salah A, Li K, Yin L (2021) A whale optimization system for energy-efficient container placement in data centers. Expert Syst Appl 164:113719
22. Mohanapriya N, Kousalya G, Balakrishnan P, Pethuru Raj C (2018) Energy efficient workflow scheduling with virtual machine consolidation for green cloud computing. J Intell Fuzzy Syst 34(3):1561–1572
23. N. Garg, M. Raj, I. Gupta, V. Kumar, and G. Sinha, "Energy-efficient scientific workflow scheduling algorithm in cloud environment," *Wireless Communications and Mobile Computing,* vol. 2022, 2022.

Publisher's Note