



OPEN

An optimal workflow scheduling in IoT-fog-cloud system for minimizing time and energy

Roqia Rateb¹, Ahmed Adnan Hadi², Venkata Mohit Tamanampudi³, Laith Abualigah^{4,8}✉, Absalom E. Ezugwu⁵✉, Ahmed Ibrahim Alzahrani⁶, Fahad Alblehai⁶ & Heming Jia⁷

Today, with the increasing use of the Internet of Things (IoT) in the world, various workflows that need to be stored and processed on the computing platforms. But this issue, causes an increase in costs for computing resources providers, and as a result, system Energy Consumption (EC) is also reduced. Therefore, this paper examines the workflow scheduling problem of IoT devices in the fog-cloud environment, where reducing the EC of the computing system and reducing the MakeSpan Time (MST) of workflows as main objectives, under the constraints of priority, deadline and reliability. Therefore, in order to achieve these objectives, the combination of Aquila and Salp Swarm Algorithms (ASSA) is used to select the best Virtual Machines (VMs) for the execution of workflows. So, in each iteration of ASSA execution, a number of VMs are selected by the ASSA. Then by using the Reducing MakeSpan Time (RMST) technique, the MST of the workflow on selected VMs is reduced, while maintaining reliability and deadline. Then, using VM merging and Dynamic Voltage Frequency Scaling (DVFS) technique on the output from RMST, the static and dynamic EC is reduced, respectively. Experimental results show the effectiveness of the proposed method compared to previous methods.

Keywords Fog-cloud computing, Workflow scheduling, Dynamic Voltage Frequency Scaling, Aquila-Salp Swarm Algorithm

Internet of Things (IoT) refers to physical devices that are located at the edge of the IoT-fog-cloud network and are connected to the internet^{1,2}. In cloud computing data centers, there are a number of servers, and on each server, there are a large number of resources or Virtual Machines (VMs) that are used by IoT based on proper management^{3–7}. But they, the use of cloud resources is associated with long delays in response time due to the high geographical distance they have from network edge devices, which may cause problems for some IoT device applications such as workflows^{8–12}. The fog layer is an intermediate layer between the cloud layer and IoT devices that can reduce the response delay^{13–17}. The cloud and fog layers consist of a combination of different resources, and the fog layer is generally less capable than the cloud layer^{18–22}. Scheduling is one of the practical approaches to efficient resource management in the fog-cloud system as virtual, which is used to allocate a set of requests by the IoT to the most suitable sources in fog-cloud nodes^{23–27}. Also, it considers the specified constraints by different users. The rapid increase in the computing requests of various users around the world leads to an increase in Energy Consumption (EC) in fog/cloud nodes, which causes rising costs.

Consequently, reducing the EC is very important for researchers^{28–32}. In addition to reducing EC, being Real-Time (meeting the deadline and not missing it) of task, and reliability are two essential services that the cloud and fog computing nodes must provide. Being Real-Time is the response time of a task/workflow with the objective of minimizing task completion time or MakeSpan Time (MST). Also, by reducing the MST of workflows, Virtual Machines (VMs) spend less time executing workflows, and as a result, system EC is also reduced. Being Real-Time can be achieved by increasing the frequency of the VM's CPU. Also, reliability is the degree to which a task or workflow is successfully executed^{33–37}. It isn't easy to balance the three metrics of reducing the MST of

¹Department of Computer Science, College of Information Technology, Al-Ahliyya Amman University, Amman, Jordan. ²Artificial Intelligence Sciences Department, College of Sciences, Al-Mustaqbal University, 51001 Babil, Iraq. ³JPMorgan Chase, New York, USA. ⁴Computer Science Department, Al Al-Bayt University, Mafraq 25113, Jordan. ⁵Unit for Data Science and Computing, North-West University, 11 Hofman Street, Potchefstroom 2520, South Africa. ⁶Computer Science Department, Community College, King Saud University, 11437 Riyadh, Saudi Arabia. ⁷School of Information Engineering, Sanming University, Sanming 365004, China. ⁸Centre for Research Impact & Outcome, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura 140401, Punjab, India. ✉email: aligah.2020@gmail.com; Absalom.ezugwu@nwu.ac.za

workflow tasks, increasing the reliability, and reducing the EC of VMs. Increasing the reliability of VMs in a fog-cloud computing system can increase the EC to execute a workflow. EC consists of two parts^{4,38–40}:

1) dynamic energy due to switching activities and static energy due to leakage current. Dynamic energy can be declined by reducing the VM frequency, which is also called Dynamic Voltage and Frequency Scaling (DVFS). The DVFS method can be applied to the CPU of all computing systems, including mobile devices, cloud data centers, etc.^{4,41–44}. Therefore, each VM executes on a physical CPU equipped with DVFS, which works at different voltage and frequency levels^{4,45,46}. The DVFS method reduces the operating frequency and voltage of the corresponding VM's CPU to reduce the CPU EC during the execution of the workflow⁴¹. However, reducing the VM's CPU frequency increases the execution time of the mapped task to the VM, and as a result, the initial deadline of tasks and workflows may be missed. Therefore, the VM's CPU frequency should be adjusted using DVFS to reduce the MST of the workflow, maintain the workflow deadline, maintain reliability, and reduce EC^{4,41}. Task/workflow scheduling problems in a fog-cloud environment are NP-hard problems^{42–45}. Due to the importance of EC, cost, reliability, deadline, and response time metrics in a fog/cloud system, we will briefly explain a number of task scheduling and workflow scheduling techniques that are discussed on these metrics and have used various MH algorithms. Many issues in the field of network types have been solved without optimization and optimization^{46–50}. Certainly, using different optimization algorithms to solve the problem can help find the best solutions^{51–55}. Different algorithms have also been used in a variety of problems to optimize the results^{56–60}.

The rest of the paper is organized as follows: in “Related works” section, the related works are described. “System model, assumptions, and problem formulation” section describes the system model, assumptions, and how to formulate the problem. In “The proposed workflow scheduling method” section, the proposed method and all algorithms, prerequisites, and techniques used in the proposed method are completely described. Section 5 describes the evaluation metrics experimental results and compares the proposed method with some other methods, and Section 6 includes the conclusion.

Related works

In this section, a number of papers have been first reviewed in the field of task/workflow scheduling in fog and cloud environments using MH algorithms. EC, cost, MST, deadline, and reliability metrics are then briefly described.

Metaheuristic algorithms for task scheduling

Several MH algorithms have been explored in the context of fog and cloud environments to address energy consumption and task scheduling issues.

- The authors proposed a method based on DVFS and IWO-CA (a combination of Invasive Weed Optimization and Cultural Evolution Algorithm) to reduce energy consumption by scheduling tasks at lower voltages and frequencies. The focus was on task priority constraints in fog environments⁶¹.
- The authors used a DVFS-based Genetic Algorithm (GA) to minimize energy consumption while considering deadline constraints. Their approach employed the Minimizing Scheduling Length (MSL) technique to reduce workflow scheduling time⁶².
- Authors utilized Ant Colony Optimization (ACO) to optimize CPU execution time for workflow scheduling in fog environments, focusing primarily on reducing makespan⁶³.
- The authors proposed a cost-aware workflow scheduling method using Particle Swarm Optimization (PSO). Their algorithm considered cost, communication delay, and reliability as key metrics⁶⁴.

Energy-efficient scheduling techniques

Energy-efficient task scheduling remains a critical challenge in cloud and fog environments. The following studies utilized different methods to reduce energy consumption:

- The authors introduced a Multi-Objective Genetic Algorithm (MOGA) that optimized workflow scheduling by minimizing both makespan and energy consumption⁶⁵.
- The authors developed the AEOSSA algorithm, combining Artificial Ecosystem-based Optimization (AEO) and Salp Swarm Algorithm (SSA), to minimize makespan while optimizing task requests in fog-cloud environments¹⁹.
- The authors proposed the DMFO-DE algorithm, which combined Moth-Flame Optimization (MFO) and Differential Evolution (DE) to enhance convergence speed and minimize cost, energy consumption, and makespan⁴.
- The authors developed a task-aware scheduling method using DVFS, focusing on reducing energy consumption and makespan in fog environments⁶⁶.

MakeSpan optimization in fog and cloud environments

Optimization of makespan, along with other metrics such as cost and energy consumption, is an essential aspect of efficient task scheduling. The following works targeted makespan reduction:

- Authors used the Firefly Algorithm (FA) to optimize workflow scheduling by focusing on makespan, reliability, and deadline constraints⁶⁷.
- Authors applied DVFS in vehicle physical systems to improve reliability, reduce energy consumption, and minimize makespan⁶⁸.

- A novel combination of Whale Optimization Algorithm (WOA) and Aquila Optimizer (AO) was proposed by Salehnia et al. to optimize makespan in cloud environments for IoT task scheduling⁶⁹.
- Multi-objective MFO Algorithm proposed by Salehnia et al. aimed at reducing energy consumption, makespan, and improving throughput time by considering optimal virtual machines (VMs) for task scheduling⁷⁰.

The relevant works are listed in Table 1 and the strengths and weaknesses of each are also specified.

As illustrated in the table, the proposed method differs from existing approaches by simultaneously addressing energy consumption, makespan, and task priorities using a multi-phase, logical scheduling approach. While existing methods tend to focus on one or two of these objectives, our method leverages a holistic approach that integrates energy efficiency, reliability, and task deadlines into the scheduling process.

Comparative analysis of existing methods

The table below provides a comparative analysis of the key features of the discussed algorithms and how they address task scheduling objectives in fog and cloud environments.

The mentioned methods in the related work section did not control the MST values and energy consumption in a completely logical and two-phase manner at the same time. Also, they did not consider the constraints of priority, deadline, and reliability. Some related works have only focused on reducing energy consumption and task priority constraints, some have focused on reducing energy consumption and MST by maintaining task deadlines, some have focused on reducing MST time, some have focused on reducing energy consumption and observing the constraints of time and reliability, some have checked cost, communication delay, and reliability metrics, some have checked cost, MST, and EC metrics, some have checked MST and EC metrics, some have checked have focused on reducing energy consumption by considering reliability and MST constraints, and some have focused on EC, MST and throughput time.

To address the gaps mentioned above, this paper examines the workflow scheduling problem of IoT devices in the fog-cloud environment, where reducing the EC of the fog-cloud computing system and reducing the MST of workflows as the main objectives under the constraints of priority, deadline, and reliability. This paper proposes a novel workflow scheduling method that optimizes energy consumption and MakeSpan Time in IoT-Fog-Cloud systems by integrating the Aquila Optimizer and Salp Swarm Algorithm (ASSA) with advanced techniques like Dynamic Voltage Frequency Scaling (DVFS) and VM merging. In order to achieve these objectives and maintain priority, therefore, deadline and reliability restrictions, our contributions are:

1. The combination of Aquila and SSA (ASSA) is used to select the best VMs for the execution of workflows by optimizing the parameters. Because according to the literature, the development and design of a strong MH algorithm that is suitable for scheduling in heterogeneous fog-cloud computing systems and can perform better than the algorithms that were mentioned in the literature, is a challenging issue.
2. SSA⁷¹ is an MH algorithm that is a little weak in the exploration phase. Therefore, in this paper, the exploration phase of Aquila Optimizer (AO)⁷² is used as an alternative to the exploration phase of SSA. So, for each iteration that the ASSA is executed, a number of VMs are selected by the ASSA,
3. Then, by using the technique of Reducing MakeSpan Time (RMST), the MST of the workflow is reduced on the selected VMs while maintaining priority, reliability, and deadline.
4. Then, using VM merging and Dynamic Voltage Frequency Scaling (DVFS) technique on the output from RMST, that is, the Actual Finish Time (AFT) of workflows on the VMs, the static and dynamic EC are reduced, respectively.
5. In ASSA, the objective function is a combination of RMST, DVFS, and VM merging. ASSA finally selects the most optimal VMs to perform workflows by minimizing the objective function.
6. We compare our proposed method with other methods. ASSA achieves better performance in terms of EC and MST while maintaining priority, reliability, and deadline than DVFS⁴¹, AO⁷², SSA⁷¹, DE-MFO⁴, PSO⁷³, FA⁷⁴, Harris Hawks Optimization (HHO)⁷⁵. By using the RMST, DVFS technique, and selected VM merging technique, ASSA can reduce EC and MST while maintaining required constraints, including deadline, priority, Real-Time, and reliability.

The summary of the main notations used is listed in Table 2.

Algorithm	Primary objective	Constraints	Optimization technique	Strengths	Weaknesses
IWO-CA (Hosseinioun et al.)	Energy Consumption	Task Priority	DVFS + IWO-CA	Efficient in EC reduction	Limited scope in makespan optimization
GA (Meena et al.)	Energy Consumption	Deadline	DVFS + GA	Effective for deadline-sensitive tasks	Lack of cost analysis
ACO (Mahmud and Buyya)	Makespan	CPU Execution Time	ACO	Good CPU utilization	Limited EC reduction
PSO (Ding et al.)	Cost & Reliability	Communication Delay	PSO	Comprehensive metric consideration	Complex to implement
MOGA (Wang et al.)	Makespan & EC	Multi-Objective	MOGA	Balances EC and makespan	High computational overhead
AEOSSA (Elaziz et al.)	Makespan	Task Requests	AEO + SSA	Combines exploration and exploitation	Limited focus on energy consumption

Table 1. Strengths and weaknesses of related work.

Notations	Definition	Notations	Definition
W	A DAG workflow	$t_{i,k}$	Time interval of t_j to vm_k
$Edge_{ij}$	Communication cost between the tasks of i -th and j -th in the workflow w	$pred(t_j)$	Predecessors of task t_j
t_i	i -th task in workflow w	λ_k	Constant failure rate per time unit of a vm_k
vm_k	k -th virtual machine	α_j	Circuit-dependent constant
$f_{j,k}$	Frequency t_j on the vm_k	$et_{j,k}$	Execution time or the calculation time of the j -th task in workflow on the k -th VM
m_j	Dynamic power exponent	st_j	Start time of the task t_j
α_j	Circuit-dependent constant	ft_{t_j, f_k}	Finish time of task t_j assigned on vm_k at frequency level f_k
α_r	Reliability ratio	st_{t_j, f_k}	Start time of task t_j assigned on vm_k at frequency level f_k
$avail[vm_k]$	Earliest Available Time (EAT) of vm_k in order to execute the desired task	et_{worst}	Worst execution time of task t_j on vm_k at frequency f_k
aet_j	Mean computation time of task t_j on all VMs	$succ(t_j)$	Successors of task t_j
$Rel_{max}(w)$	Maximum reliability of a workflow w	$rank_u(t_j)$	Upward rank value of a task
Rel_{max}^j	Rel_{max}^j is maximum reliability of j -th task	$t_{o(j)}$	Task with priority j
$R_{Req}^{o(j)}$	Required reliability of task $t_{o(j)}$	$RT_{req}(t_j, vm_k)$	MST requirement of the task t_j on the VM's CPU vm_k
$EST(t_j, vm_k)$	Earliest Start Time (EST) of task t_j on vm_k	$AFT(t_j, vm_k)$	Actual Finish Time (AFT) of task t_j on vm_k
t_{entry}	input task in the workflow w	t_{exit}	Output task in the workflow w
$AST(t_{i1})$	Actual Start Time (AST) of task t_{i1}	$\Lambda[vm_k]$	End spot of the available time of vm_k

Table 2. The summarization of main notations.

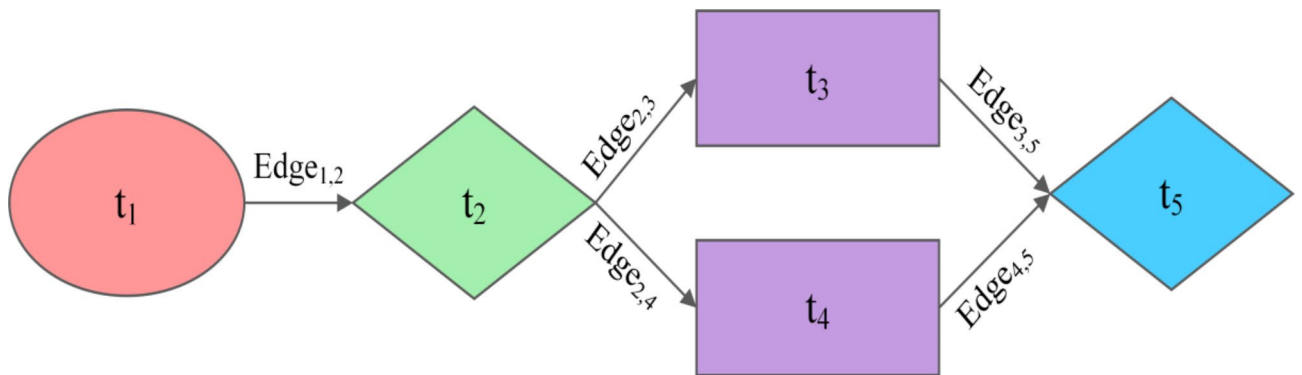


Fig. 1. Model of a workflow.

System model, assumptions, and problem formulation

This section describes the system model in the workflow scheduling process, as well as the mathematical formulation of the workflow scheduling problem in the fog-cloud network, as well as the assumptions that are considered.

System model and related assumptions

A workflow w (Task, Edge) is represented as a Directed Acyclic Graph (DAG), which consists of a set of dependent tasks. The Task represents the set of tasks in the workflow w , and Edge represents the set of directed edges that exist between the existing tasks and specify the dependency or communication cost between the tasks in the workflow w ⁴¹. For example, $Edge_{1,2}$ represents the dependency or communication cost between task 1 and task 2. Task t_1 is the predecessor of task t_2 , and task t_2 is the successor of t_1 . Therefore, in a workflow w , all the predecessor tasks in the workflow are completed and executed before the successor tasks start processing⁴. Figure 1 shows the workflow structure.

There are assumptions for workflow scheduling in the fog-cloud system:

1. There is a set of workflows for sending to fog-cloud nodes.

$$W = \{w_1, w_2, \dots, w_{N1}\} \quad (1)$$

2. Each workflow w_i ($i=1, 2, \dots, N1$) consists of a number of tasks⁴.

$$w_j = \{t_1, t_2, \dots, t_{N2}\}, j = 1, 2, \dots, N1 \quad (2)$$

3. Each workflow has an execution time matrix in the form of ET, and in the matrix ET, where $et_{j,k}$ ($j=1, 2, \dots, N2$ and $k=1, 2, \dots, Mn$) represents the execution time or the calculation time of the j -th task in workflow on the k -th VM with the maximum frequency⁴.
4. The fog-cloud environment consists of $M1$ $Node_{cloud}$ which is defined as Eq. (3)⁶³:

$$Node_{cloud} = \{Node_1, Node_2, \dots, Node_{M1}\} \quad (3)$$

In Eq. (3), $Node_z$ ($z=1, \dots, M1$) represents the z -th server in the fog-cloud environment.

5. A fog-cloud node can execute several separate heterogeneous VM samples, and each VM sample includes various resources such as CPU and memory⁶⁷.
6. There are several VMs on each $Node_z$.

$$\begin{aligned} Node_1 &= \{vm_{1,1}, vm_{1,2}, \dots, vm_{1,M2}\} \\ Node_2 &= \{vm_{2,1}, vm_{2,2}, \dots, vm_{2,M3}\} \\ Node_{M1} &= \{vm_{M1,1}, vm_{M1,2}, \dots, vm_{M1,k}\} \end{aligned} \quad (4)$$

The number of resources or VMs on different nodes is not the same.

7. The VMs are heterogeneous.
8. Each VM is executed in a physical CPU equipped with DVFS, which works at different frequencies/voltages, and it is formulated with three values (f, v, c), that indicate the frequency, voltage, and maximum processing capacity of the CPU, respectively^{4,67}.
9. The processing capacity of the VM's CPU is measured over a Million Instructions Per Second (MIPS)^{4,76–78}.
10. For each workflow w , Edge is a $N2 \times Mn$ communication cost matrix, where $Edge_{i,j}$ denotes the dependency and communication cost between task t_i and t_j in workflow. So, $Edge_{i,j} > 0$ if t_i must be completed ahead of t_j , and $Edge_{i,j} = 0$ otherwise⁷⁶.
11. Each task in a DAG is mapped on the corresponding VM with a certain frequency. $f_{j,k}$ is used to indicate the task execution frequency t_j on the vm_k . Therefore, the total EC of the task t_j is equal to^{67,76}:

$$Energy(t_j, vm_k, f_k) = (Energy_{static_k} + Energy_{dynamic}) \times \frac{f_{k,max}}{f_k} \times et_{j,k} \quad (5)$$

$$Energy_{dynamic} = \alpha_j \cdot f_k^{m_j} \quad (6)$$

where, $f_{k,max}$ is the max frequency of vm_k , $Energy_{static_k}$ stands for quiescent EC of vm_k due to leakage current. $Energy_{dynamic}$ for dynamic EC due to switching activities for execute the task t_j on the vm_k , which depends on the VM's CPU frequency and m_j represents dynamic power exponent (usually assumed that $m_j > 2$)⁷⁶. Where, α_j is a constant and f_k is the CPU frequency of the vm_k .

12. A CPU can take any frequency value between f_{min} and $f_{max} = 1$. The computation time of each task in workflow w is measured based on f_{max} ⁷⁶.
13. The total EC of the workflow w ($Energy_{total_w}$) is equal to the EC of all the tasks in workflow w , that is⁷⁶:

$$Energy_{total_w} = \sum_{j=1}^{N2} Energy(t_j, vm_k, f_k), k \in Mn \quad (7)$$

14. Reliability means the possibility that the system will always work correctly and perform its tasks well in a certain period despite possible failures^{67,79}.
15. Only the failure caused by the execution of the task is discussed.
16. For a DVFS-based system, the failure rate depends on the VM's CPU frequency, according to^{80,81}. The failure rate of vm_k with frequency f_k is equal to^{82–84}:

$$\lambda_k = \lambda_{k,max} \times 10^{\frac{d \times (f_{k,max} - f_k)}{f_{k,max} - f_{k,min}}} \quad (8)$$

where, λ_k is the failure rate per time unit of a vm_k , and $\lambda_{j,max}$ is the failure rate per unit time of vm_k with the maximum frequency, $f_{k,max}$ and $f_{k,min}$ is the minimum frequency and maximum frequency of the k -th VM's CPU, respectively, and d is a constant.

17. Directly, the reliability of executing task t_j on vm_k with frequency f_k is as^{80,81,84}:

$$Rel(t_j, vm_k, f_k) = e^{-\lambda_k \times \frac{t_{j,k} \times f_{k,max}}{f_k}} = e^{-\lambda_{k,max} \times 10^{\frac{d \times (f_{k,max} - f_k)}{f_{k,max} - f_{k,min}} \times \frac{t_{j,k} \times f_{k,max}}{f_k}}} \quad (9)$$

where, $t_{j,k}$ is the time interval of t_j to vm_k . So, the level of reliability decreases uniformly with the decrease in the VM's CPU frequency. Therefore, it can be concluded that reducing the voltage and frequency values to save the system energy may reduce the reliability value.

18. The total reliability of the workflow w is^{80,81,84}:

$$Rel_{total_w} = \sum_{j=1}^{N2} Rel(t_j, vm_k, f_k), \quad k \in Mn \quad (10)$$

Problem formulation

Considering the requirements of maintaining reliability and deadline for a Real-Time workflow w , this paper tries to reduce MST and EC by using the combination of ASSA, RMST and DVFS techniques and VM merging. A set $(b_{j,k}, st_j, f_k)$ is used to determine the status of each task. $b_{j,k} = 1$ if task t_j is mapped to vm_k , otherwise, $b_{j,k} = 0$. st_j indicates the start time of the task t_j , and k indicates the VM frequency value for the execution of t_j ^{67,76}. So, for each task t_j and for each vm_k :

$$st_j \geq \max_{t_j \in pred(t_j)} \left(st_q + \sum_{k=1}^{Mn} \left(b_{q,k} \cdot et_{q,k} \cdot \frac{1}{f_q} + \frac{1}{2} |b_{q,k} - b_{j,k}| \cdot Edge_{q,j} \right) \right) \quad (11)$$

where, $pred(t_j)$ represents the predecessors of task t_j , and $q < j$, and t_q is the predecessor of the t_j . According to Eq. (11), the dependence of the task on the previous tasks in the workflow is considered. Therefore, a task starts executing only if all the previous tasks in the workflow have been executed and completed and all the results of the previous tasks in the workflow have been received.

$$\left(\sum_{k=1}^{Mn} b_{i1,k} \cdot b_{i2,k} \right) \cdot \left(st_{i1} + \sum_{k=1}^{Mn} b_{i1,k} \cdot et_{i1,k} \cdot \frac{1}{f_{i1}} - st_{i2} \right) \cdot \left(st_{i2} + \sum_{k=1}^{Mn} b_{i2,k} \cdot et_{i2,k} \cdot \frac{1}{f_{i2}} - st_{i1} \right) \quad (12)$$

where, $i1$ and $i2 \in N2$. According to Eq. (12), task execution in workflow is not allowed to be preempted.

$$st_j + \sum_{k=1}^{Mn} b_{j,k} \cdot et_{j,k} \cdot \frac{1}{f_j} - st_j \leq deadline_{Req} \quad (13)$$

where, $deadline_{Req}$ is a minimum deadline that a workflow must complete before it is missed. Equation (13) describes the workflow scheduling requirements to complete the workflow w within the given deadline and before the deadline is missed. Equation (14) describes the reliability condition. That is, the reliability of the workflow (Rel_{total_w}) should not be lower than the minimum reliability.

$$Rel_{total_w} \geq Rel_{Req} \quad (14)$$

where, Rel_{Req} is a minimum reliability that a workflow must complete.

The proposed workflow scheduling method

In this section, the proposed workflow scheduling algorithm will be described. Therefore, every iteration that ASSA is executed:

A number of VMs are randomly selected from the heterogeneous set of VMs that are located on the nodes of the fog-cloud environment. Then, without considering the problem of reducing EC in the fog-cloud environment, the RMST algorithm is executed in order to reduce the MST (completion time) for the execution of the corresponding workflow on the selected VMs by ASSA. The RMST output is the AFT of the task t_j on vm_k , is used on the selected VMs to check whether it is possible to meet all the deadline, priority, and reliability constraints. In the last step, after ensuring that all the constraints are met, VM merging and DVFS techniques are performed to reduce EC (static and dynamic). In ASSA, the objective function is a combination of RMST, DVFS, and VM merging. ASSA finally selects the most optimal VMs to perform workflows by minimizing the objective function. Figure 2 shows the Diagram of the proposed workflow scheduling method using ASSA and RMST, VMM, and DVFS techniques.

The objective function is calculated using Eq. (15).

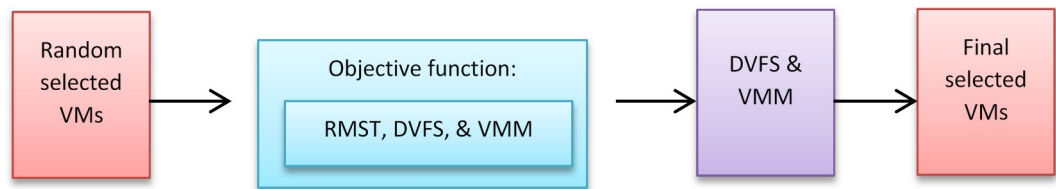


Fig. 2. Diagram of the proposed workflow scheduling method using ASSA and RMST, VMM and DVFS techniques.

$$Fit = \frac{c_1 \times Energy_{total_w} + c_2 \times Makespan_{total_w}}{Rel_{total_w}} \quad (15)$$

where, c_1 and c_2 are two fixed coefficients that are used to normalize the parameters and are equal to 0.001, $Makespan_{total_w}$ represents the amount of total MST of workflow w , and is calculated as follows⁴:

$$Makespan_{total_w} = \max(st_{t_j, f_k} + et_{worst}) \quad (16)$$

where, st_{t_j, f_k} and ft_{t_j, f_k} represents the start and finish time of the task t_j assigned on vm_k at frequency level f_k , respectively, and et_{worst} is the worst execution time of task t_j on vm_k at frequency f_k .

$$st_{t_j, f_k} = \begin{cases} 0, & \text{if } t_j \text{ is a entry task} \\ \max(ft_{t_j, f_k}), & \text{otherwise} \end{cases} \quad (17)$$

Therefore, the MST of workflow w equals the finish time of the last task in the workflow w . Next, each step of the proposed method will be described.

First phase: ASSA in a workflow scheduling problem

The ASSA is designed using the combination of AO and SSA. In the following sub-sections, AO, SSA, and ASSA will be described.

Aquila optimizer algorithm

The AO algorithm⁷² is a Swarm Intelligence algorithm in which there are four hunting behaviors of Aquila for different types of prey. AO searches the search space to find the best solutions globally and locally by using exploration and exploitation phases and finally converges to the final optimal solution. A brief description of the mathematical model of the AO algorithm and its algorithm steps is as follows:

Step 1: Extensive exploration.

The mathematical representation of this behavior is written as follows:

$$A(t+1) = A_{best}(t) \times \left(1 - \frac{t}{T}\right) + (A_{avg}(t) - A_{best}(t) \times rand) \quad (18)$$

$$A_{avg}(t) = \frac{1}{N} \sum_{k=1}^N A_k(t) \quad (19)$$

where $A_{best}(t)$ represents the best position achieved so far, and $A_{avg}(t)$ represents the mean position of all Aquilas in the current iteration, t and T are the current iteration and the maximum number of iterations, respectively, N is the population size, and $rand \in [0, 1]$.

Step 2: Narrow exploration.

The position update equation is displayed as follows:

$$A(t+1) = A_{best}(t) \times LF(D) + A_R(t) + (y - x) \times rand \quad (20)$$

where $A_R(t)$ represents a random position of the Aquila, and D is the dimension size, $LF(D)$ represents the Levy flight function, which is given as:

$$LF(D) = s \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}} \quad (21)$$

$$\sigma = \left[\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right] \quad (22)$$

where s and β are constant values equal to 0.01 and 1.5, respectively, u and $v \in [0, 1]$, y and x are used to display the spiral shape in the search, which is as:

$$\begin{cases} x = r \times \sin(\theta) \\ y = r \times \cos(\theta) \\ r = r_1 + 0.00565 \times D_1 \\ \theta = -w \times D_1 + \frac{3 \times \pi}{2} \end{cases} \quad (23)$$

where, r_1 means the number of search cycles between 1 and 20, D_1 consists of integers from 1 to the dimension size (D) and is equal to 0.005.

Step 3: Extensive exploitation.

This behavior is presented as follows:

$$A(t+1) = (A_{best}(t) - A_{avg}(t)) \times \alpha - rand + ((UB - LB) \times rand + LB) \times \delta \quad (24)$$

where α and δ are the exploitation adjustment parameters that are fixed at 0.1, UB and LB are the upper and lower bounds of the problem.

Step 4: Limited exploitation.

The mathematical representation of this behavior is as follows:

$$A(t+1) = QF \times A_{best}(t) - (G_1 \times A(t)) \times rand - G_2 \times LF(D) + rand \times G_1 \quad (25)$$

$$QF(t) = t^{\left(\frac{2 \times rand - 1}{(1-T)^2}\right)} \quad (26)$$

$$G_1 = 2 \times rand - 1 \quad (27)$$

$$G_2 = 2 \times \left(1 - \frac{t}{T}\right) \quad (28)$$

where $A(t)$ is the current position, $QF(t)$ represents the value of the quality function used to balance the search strategy, G_1 shows the movement parameter of the Aquila when tracking the prey, which is a random number between $[-1, 1]$, G_2 shows the flight slope when chasing prey, which decreases linearly from 2 to 0. After repeated iterations of the algorithm, and the algorithm reaches the stopping condition, the AO output is obtained.

Salp swarm algorithm

SSA is an MH algorithm⁷¹ where, at the beginning of optimization, a set of solutions is randomly generated and distributed over the search space. Then, the objective function value is calculated for each search agent (salp/solution). The solutions are divided into two categories of leaders and followers according to the objective function value. The workflow scheduling problem is solved by minimizing the objective function, so the salps with the lowest objective function value are determined as leaders, and the salps with the highest objective function value are considered as followers. In SSA, each solution (Salps) $\vec{S}_k$ ($k = 1, \dots, N$) is represented as a vector $1 \times N$ according to Eq. (30) where S_k represents the index of the k -th VM.

$$\vec{S}_i = (s_1, s_2, \dots, s_i) \quad (29)$$

$$\vec{S}_k = \begin{cases} F_k + c_1((UB - LB)r_1 + LB) & r_2 < 0.5 \\ k - c_1((UB - LB)r_1 + LB) & r_2 \geq 0.5 \end{cases} \quad (30)$$

where, r_1 and $r_2 \in [0, 1]$, F_k represents food source in the k -th dimension, $\vec{S}_k$ represents each solution (the new position of the leader) in the k -th dimension. In order to balance the exploration and exploitation phases, c_1 is used and calculated as Eq. (31).

$$c_1 = 2e^{-\left(\frac{4 \times iter}{iter_{max}}\right)^2} \quad (31)$$

where, $iter$ and $iter_{max}$ represent the current iterations and the total number of iterations. After randomly generating and distributing the solutions, the position of each solution is updated according to its type (i.e. leader or follower). Equation (32) is used to update the position of the follower solutions.

$$S_k = \frac{1}{2}a \cdot iter^2 + v_0 \cdot iter \quad (32)$$

where, S_k represents the position of the k -th follower solution, a and v_0 represent the acceleration and the velocity, respectively ($\Delta iter = 1$ and $v_0 = 0$). Therefore, it is updated as Eq. (33):

$$S_k = \frac{1}{2} (S_k + S_{k-1}) \quad (33)$$

The new solutions are evaluated and updated, and in each iteration, the best solution is determined based on the objective function. This process continues until the stop condition is met. Finally, the output of the SSA will be obtained.

Workflow scheduling in fog-cloud environment using ASSA

In order to solve the workflow scheduling problem, in this paper, the exploration phase of AO is combined with the exploitation phase of SSA. The ASSA algorithm is obtained, which is stronger than AO and SSA and can provide better scheduling for the execution of IoT device applications, which are the same workflows, and select the most suitable VMs available in the fog-cloud environment to perform the tasks of each workflow. Therefore, in ASSA, at the beginning of the optimization, an initial population of solutions, including N solutions, is randomly created by AO and distributed over the search space (set of VMs in the fog-cloud environment). This process is done by setting the initial value for a set of N solutions and converting them into integer solutions. The task scheduling problem in the fog-cloud environment is an integer problem, which is calculated according to Eq. (34).

$$x_k = rand(0, 1) \times (UB - LB) + LB \quad k = 1, 2, \dots, N \quad (34)$$

where, LB and UB in this research are equal to 1 and the number of the VMs available on all servers, respectively. Then, the objective function value for all solutions is calculated using Eq. (15).

Therefore, according to Eq. (18) and Eq. (20), the stages of wide exploration and narrow exploration are performed from AO. The AO is merged into the SSA as the leader role, and the exploitation phase of the SSA is maintained. Then the exploitation steps of SSA are done using Eq. (33). In general, an optimal mapping of a set of workflow tasks on a set of VMs is performed using the proposed ASSA. Figure 3 shows the diagram of ASSA. The position of the solutions, which is in the form of a vector, indicates the index of the VMs available on the fog-cloud nodes, in which case the size of the dimensions of each solution is equal to the number of tasks, and their values indicate the index of the corresponding VM, which assigned to the desired task. In other words, the goal is to determine the most efficient VMs for executing tasks using ASSA so that the MST in the system is reduced, reliability is maintained, system EC is reduced, and the priority and deadline constraints of tasks and workflows are reduced. So:

1. The combination of Aquila and SSA (ASSA) is used to select the best VMs for the execution of workflows by optimizing the parameters. Because, according to the literature, the development and design of a strong MH algorithm that is suitable for scheduling in heterogeneous fog-cloud computing systems and can perform better than the algorithms that were mentioned in the literature is a challenging issue.
2. SSA is an MH algorithm that is a little weak in the exploration phase. Therefore, in this paper, the exploration phase of Aquila Optimizer (AO) is used as an alternative to the exploration phase of SSA. So, for each iteration that the ASSA is executed, a number of VMs are selected by the ASSA,
3. Then, by using the technique of Reducing MakeSpan Time (RMST), the MST of the workflow is reduced on the selected VMs while maintaining priority, reliability, and deadline.
4. Then, using VM merging and Dynamic Voltage Frequency Scaling (DVFS) technique on the output from RMST, that is, the Actual Finish Time (AFT) of workflows on the VMs, the static and dynamic EC are reduced, respectively.

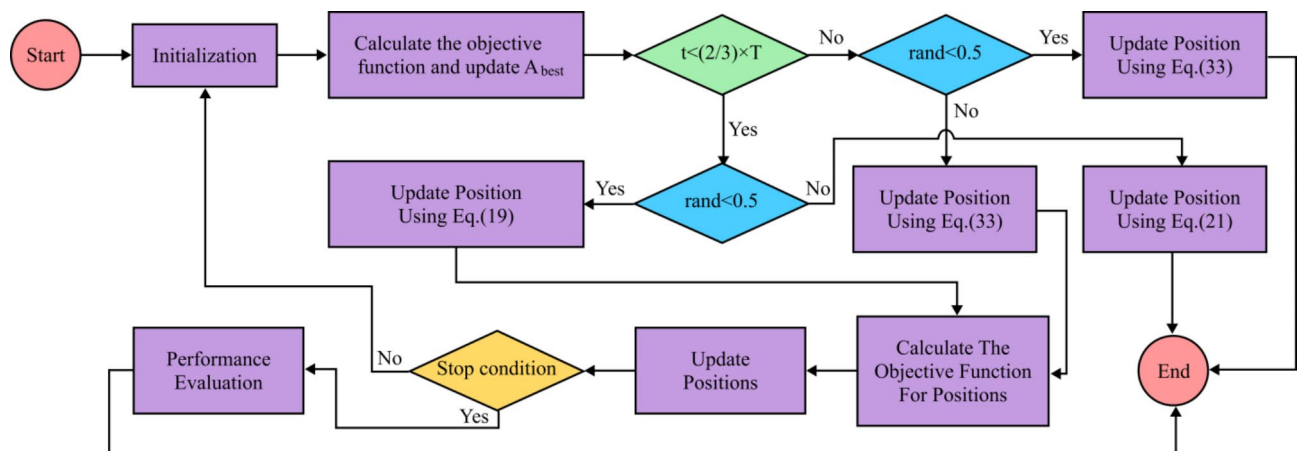


Fig. 3. ASSA diagram.

5. In ASSA, the objective function is a combination of RMST, DVFS, and VM merging. ASSA finally selects the most optimal VMs to perform workflows by minimizing the objective function.

For example, if 4 nodes in the fog-cloud environment have 3, 5, 6, and 4 VMs, respectively, and there is a workflow with 8 tasks, the dimensionality of each solution is equal to an 8-member vector. The value of each dimension is equal to the index of one of the VMs on each node in the fog-cloud environment.

$$\begin{aligned} Node_1 &= \{vm_{1,1}, vm_{1,2}, vm_{1,3}\} \\ Node_2 &= \{vm_{2,1}, vm_{2,2}, vm_{2,3}, vm_{2,4}, vm_{2,5}\} \\ Node_3 &= \{vm_{3,1}, vm_{3,2}, vm_{3,3}, vm_{3,4}, vm_{3,5}, vm_{3,6}\} \\ Node_4 &= \{vm_{4,1}, vm_{4,2}, vm_{4,3}, vm_{4,4}\} \end{aligned} \quad (35)$$

The search space for this collection of VMs is the following vector:

$$vm_{total} = \{vm_{1,1}, vm_{1,2}, vm_{1,3}, vm_{2,1}, vm_{2,2}, vm_{2,3}, vm_{2,4}, vm_{2,5}, vm_{3,1}, vm_{3,2}, vm_{3,3}, vm_{3,4}, vm_{3,5}, vm_{3,6}, vm_{4,1}, vm_{4,2}, vm_{4,3}, vm_{4,4}\} \quad (36)$$

If the obtained value for ASSA dimensions is as shown in Fig. 4, then task t_1 is on $vm_{1,2}$, task t_2 is on $vm_{3,5}$, task t_3 is on $vm_{3,5}$, task t_4 is on $vm_{2,5}$, task t_5 on $vm_{4,3}$, task t_6 on $vm_{3,6}$, task t_7 on $vm_{2,2}$, task t_8 on $vm_{4,3}$.

As seen in Fig. 4, tasks t_5 and t_8 are competing for $vm_{4,3}$. Therefore, considering that each task has a specific priority and deadline, tasks t_5 and t_8 are executed according to their priority on $vm_{4,3}$ in the order of their priority.

The computational complexity of AO, SSA, and ASSA

In this section, the total computational complexity of AO, SSA and ASSA is presented. The computational complexity of AO, SSA, and ASSA usually relies on three rules:

Initial solutions, calculation of objective functions, and updating solutions. For N solutions, $O(N)$ is the computational complexity of the initialization processes of the solutions. For T iterations, if the dimension size of each solution (problem dimension size) is equal to D , the computational complexity of the updating solution is $O(T \times N) + O(T \times N \times D)$, which includes searching for the best positions and updating their positions are for all solutions. Accordingly, the total computational complexity of AO is equal to $O(N \times (T \times D + T))$ ⁷². In the initialization phase, the computational cost of generated positions for the SSA is $O(N \times D)$ ⁷¹. Also, In the exploitation phase, according to⁷¹, the computational cost of update positions for the SSA is $O(2 \times N \times D)$, and the computational cost of the objective function for the solution is $O(N)$ ⁷¹. The SSA⁷² has computational complexity as Eq. (37).

$$O(T \times N) + O(2 \times N \times D) + O(N \times D) \quad (37)$$

Which is equal to:

$$O(N \times (3 \times T \times D + T)) \quad (38)$$

The ASSA is designed from a combination of the AO exploration phase and the SSA exploitation phase, so, for N search agent, T iteration and D dimension have computational complexity as Eq. (39).

$$O(T \times N) + O(2 \times N \times D) + O(N \times D) \quad (39)$$

Which is equal to:

$$O(N \times (3 \times T \times D + T)) \quad (40)$$

Considering that the computational complexity of AO is equal to $O(N \times (T \times D + T))$, the computational complexity of SSA is equal to $O(N \times (3 \times T \times D + T))$, and the computational complexity of ASSA is equal to $O(N \times (3 \times T \times D + T))$, it can be seen that computational complexity of ASSA is more than AO and equal to SSA.

Second phase: reducing the MakeSpan time technique

The RMST algorithm first prioritizes all workflow tasks according to their upward rank values, then, considering reliability, assigns each task of workflow to a suitable VM in the fog-cloud environment. In the end, this algorithm finds the best scheduling that has the minimum MST for the workflow under deadline, priority, and reliability

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
1,2	3,5	1,3	2,5	4,3	3,6	2,2	4,3

Fig. 4. Showing the dimensions of each solution in ASSA.

constraints. In the following, the RMST⁶² algorithm and all its requirements are completely described. According to precedence restrictions in a workflow, the priority of assigning the tasks to VMs must be determined before assigning them to VMs so that each task is scheduled according to its priority. According to^{85–89}, the upward rank value ($rank_u$) is used to determine the priority of tasks when sending them to VMs. The upward rank value of a task is defined by the Eq. (41):

$$rank_u(t_j) = \overline{aet_j} + \max_{t_j \in succ(t_j)} \{et_{i,k} + rank_u(t_j)\} \quad (41)$$

where, for task t_j , aet_j represents the mean computation time on the VMs and $succ(t_j)$ is the successors. The priorities of the tasks are ordered according to the descending order of $rank_u$. The maximum reliability of a workflow is obtained according to the Eq. (42) and Eq. (43)^{82–84}.

$$Rel_{max}(w) = \prod_{t_j \in w} Rel_{max}^j \quad (42)$$

$$Rel_{max}^j = \max_{k=1, \dots, Mn} et^{-\lambda_k e t_{j,k}} \quad (43)$$

Considering that the level of reliability of a task decreases uniformly with the reduction of the VM's CPU frequency, creating an effective balance between these two parameters is very important. Therefore, the above equations can create this balance between the reliability and the VM's CPU frequency. Notably, if $Rel_{max}(w) < Rel_{req}$, it is impossible to satisfy the reliability requirement. So this condition should be considered. The task scheduling method is such that the tasks in the workflow are mapped on the appropriate VM in the order of their priority. If $t_{o(j)}$ represents a task with priority j , tasks are sequentially mapped from $t_{o(1)}$ to $t_{o(N2)}$ on VMs.

Suppose the set $\{t_{o(1)}, \dots, t_{o(j-1)}\}$ represents the tasks that are mapped on the VMs, the task $t_{o(j)}$ represents the task being mapped to the VM, and the set $\{t_{o(j+1)}, \dots, t_{o(N2)}\}$ represents the tasks that are not mapped to any VM. The level of reliability of the set of tasks that are mapped on VMs is known. But to obtain the minimum reliability of $t_{o(j)}$ being mapped to a VM, assume that the set of unmapped tasks has maximum reliability. Therefore, the required reliability of the mapping of the $t_{o(j)}$ is as follows^{82–84}.

$$Rel_{Req}^{o(j)} = \frac{Rel_{req}}{\prod_{k=1}^{j-1} Rel(t_{o(j)}) \prod_{k1=j+1}^{N2} Rel_{max}^{o(k1)}} \quad (44)$$

where $Rel_{Req}^{o(j)}$ represents the required reliability of task $t_{o(j)}$. Given the requirement of a reliability value for a task that is mapped on a VM, the VM that can satisfy the reliability requirement of the task and has the Earliest Finish Time (EFT)⁴ is selected. The value of the time slot for the execution of $t_{o(j)}$ on the vm_k is calculated as follows:

$$EST(t_{entry}, vm_k) = 0 \quad (45)$$

$$EST(t_j, vm_k) = \max \left(avail[vm_k], \max_{t_{i1} \in pred(t_j)} \{AFT(t_{i1}, vm_k) - et_{i1,j}\} \right) \quad (46)$$

$$AFT(t_j, vm_k) = \min_{k=1, \dots, Mn} (EST(t_j, vm_k) - et_{j,k}) \quad (47)$$

where, $EST(t_j, vm_k)$ and $AFT(t_j, vm_k)$ are used to indicate the Earliest Start Time (EST)⁶² and Actual Finish Time (AFT)⁶² of task t_j on vm_k . If t_1 is an input task, $avail[vm_k]$ represents the Earliest Available Time (EAT) of vm_k in order to execute the desired task, $et_{i,j}$ is the time to communicate between t_i and t_j (EST and AFT are obtained based on the priority of the respective tasks).

Third phase: virtual machine merging and dynamic voltage frequency scaling techniques

EC includes two dynamic and static parts, and static EC may increase due to the increase of MST. Therefore, in order to create a balance between reducing the total EC (combination of static and dynamic energy) and meeting the requirements of MST and reliability, the combination of VM merging and DVFS was used in this paper.

There are two on and off modes for each VM's CPU, so there are 2^{Mn} Possibilities in using Mn VMs. To determine which VM's be shut down to save energy, the EC effect of VMs should be compared. Therefore, a VM merging algorithm is proposed to determine the most suitable VMs to turn off, which effectively reduces the static EC by turning off the respective VM's CPU^{4,62}. The VM merging method turns off the CPU of low-power VMs as much as possible to reduce static EC⁶². It is assumed that at the beginning of work, the CPU of all VMs is turned on. If the deadline, priority, MST, and reliability requirements are not met, the VM merging fails and does not continue. Otherwise, the RMST algorithm is used to check whether the mentioned requirements are met by shutting down the VM's CPU. After RMST is applied to the VMs and the RMST verification is passed by the CPUs of the respective VMs, the total EC of the VMs is calculated. After one round of calculation, if doing this leads to the greatest savings in the amount of EC of the VM, the respective VM's CPU will be shut down.

This process continues until all respective VMs are checked and no VM can be shut down without violating the mentioned requirements. At the end of the work, the total EC is obtained.

Given that, some slack allows the CPU frequency of the VM to decrease during scheduling. In the desired heterogeneous fog-cloud computing system, EC in VMs can be reduced by using the DVFS technique and reducing the CPU frequency of VMs^{4,62}. By reducing the VM CPU frequency, the amount of dynamic EC can be reduced. However, by reducing the VM's CPU frequency, the execution time and MST increase, and the reliability level decreases (according to Eq. (9)). On the other hand, increasing the MST causes the static EC to increase. Therefore, the DVFS technique should be used in a way that balances the metrics of reducing the total EC of the system and reducing the MST of the tasks as well as the reliability requirement^{4,62}.

Therefore, based on the obtained scheduling results from RMST, the VM's CPU frequency on the task is reduced from $t_{o(N2)}$ to $t_{o(1)}$ sequentially (by guaranteeing the MST and reliability required by the tasks). Therefore, the task mapping is set sequentially from $t_{o(N2)}$ to $t_{o(1)}$. The DVFS technique works at the task level, reducing the execution frequency of each task when it is mapped to a VM, which means it changes the processing frequency of a VM's CPU when performing different tasks in order to reduce EC. When setting up a task mapping, it is assumed that a number of tasks are mapped to VMs. To satisfy the MST requirement, a new expression $RT_{req}(t_j, vm_k)$ is defined to represent the MST requirement of the t_j on the VM's CPU vm_k , which can be defined as^{4,76}:

$$RT_{req}(t_{exit}, vm_k) = deadline_{req} \quad (48)$$

$$RT_{req}(t_j, vm_k) = \min(\Lambda[vm_k], \min_{t_{i1} \in succ(t_j)} \{AST(t_{i1}) - et'_{i1,j}\}) \quad (49)$$

where $AST(t_{i1})$ indicates the Actual Start Time (AST) of task t_{i1} , and $\Lambda[vm_k]$ indicates the end spot of the available time of vm_k . The EST of t_{i1} depends on the execution of its previous task in the workflow, which is obtained using $EST(t_j, vm_k)$. Finally, a time slot $[EST(t_j, vm_k), RT_{req}(t_j, vm_k)]$ is determined that can execute t_j without violating the precedence and MST constraints. $\Delta_{j,k} = [EST(t_j, vm_k), RT_{req}(t_j, vm_k)]$ is used to show the execution time slot that is available. Based on the scheduling obtained from RMST, if $Rel_{req} \leq Rel_{total_w}$, the reliability of some tasks can be reduced in exchange for reducing the CPU frequency of the VM. The reliability ratio α_r is defined as follows:

$$\alpha_r = \frac{\sqrt[N2]{Rel_{req}}}{\sqrt[N2]{Rel_{total_w}}} \quad (50)$$

where Rel_{total_w} is calculated based on the scheduling obtained from RMST. If $Rel_{req} \leq Rel_{total_w}$ the reliability ratio $\alpha_r < 1$. With remapping $t_{o(j)}$, the required reliability for $t_{o(j)}$ calculated as follows^{82–85}:

$$Rel_{Req, \alpha_r}^{o(j)} = \frac{Rel_{req}}{(\alpha_r)^i \cdot \prod_{i1=1}^{j-1} Rel(t_{o(i1)}) \prod_{i1=j+1}^{N2} Rel(t_{o(i1)})} \quad (51)$$

where, α is the reliability ratio. Due to reliability and free slack, the CPU frequency of the can be scaled down. First, the largest execution time ($et'_{o(j),k}$) on each VM is calculated according to the level of reliability, then the VM and its CPU frequency are adjusted in such a way that EC is reduced and the MST requirement is not be violated. Therefore, despite the reliability metric, $et'_{o(j),k}$ in the vm_k with the requirement of $Rel_{req, \alpha_r}^{o(j)}$ is calculated using the following equation^{82–85}:

$$10^{\frac{d(1-f_{o(j)})}{1-f_{k,min}}} \times et'_{o(j),k} = -\frac{\ln(Rel_{req, \alpha_r}^{o(j)})}{\lambda_k} \quad (52)$$

$$et'_{o(j),k} = \frac{et_{o(j),k}}{f_{o(j)}} \quad (53)$$

A binary search is used to obtain $f_{o(j)}$. Using the available slot $\Delta_{o(j),k}$, the task $t_{o(j)}$ mapped in the following ways^{67,85}:

$$\begin{cases} \begin{cases} f_{o(j)} = \frac{et_{o(j),k}}{et'_{o(j),k}}, \\ AST(t_{o(j)}) = RT_{req}(t_{o(j)}, vm_k) - et'_{o(j),k} \end{cases} & \text{If } |\Delta_{o(j),k}| \geq et'_{o(j),k} \\ \begin{cases} f_{o(j)} = \frac{et_{o(j),k}}{|\Delta_{o(j),k}|}, \\ AST(t_{o(j)}) = EST(t_{o(j)}, vm_k) \end{cases} & \text{If } et_{o(j),k} \leq et'_{o(j),k} \\ fail, & \text{otherwise} \end{cases} \quad (54)$$

$$|\Delta_{o(j),k}| = RT_{Req}(t_{o(j)}, vm_k) - EST(t_{o(j)}, vm_k) \quad (55)$$

where, $f_{o(j)}$ indicates the VM's CPU frequency for the task $t_{o(j)}$. First, the reliability ratio for the task $t_{o(j)}$ is calculated, and $t_{o(j)}$ is sequentially set from $t_{o(N2)}$ to $t_{o(1)}$.

For each task, all VMs are checked and tested. The VM that consumes the least energy is used to execute $t_{o(j)}$ and its CPU frequency is reduced accordingly.

The pseudocode of the proposed method for scheduling the workflow using RMST, VMM, and DVFS techniques, as well as the ASSA, is shown in Algorithm 1.

1. Aquila Salp Swarm Algorithm (ASSA)

Input: w (Task, Edge), Total VMs
Output: Optimal final VMs

- 1: Select several VMs randomly
- 2: Objective function is the combination of RMST, DVFS, and VMM
- 3: ASSA selects the optimal VMs to execute the workflow by minimizing the objective function.

1.1. Reducing MakeSpan Time (RMST)

Input: w (Task, Edge), ET , Rel_{req} , priority queue of workflow tasks
Output: MakeSpan Time

- 1: Calculate Rel_{max}^j and $Rel_{max}(w)$
- 2: If $Rel_{max}(w) < Rel_{req}$
- 3: Return false;
- 4: Else
- 5: Sort tasks in descending order of $rank_u(t_j)$ in the priority queue;
- 6: for each $t_{o(j)} \in$ priority queue;
- 7: for each $vm_k \in$ VMs;
- 8: $EST(t_{o(j)}, vm_k)$;
- 9: End for
- 10: Mapping $t_{o(j)}$ on the VM that achieves AFT ($t_{o(j)}$);
- 11: End for;
- 12: End if

2. Virtual Machine Merging (VMM)

Input: w (Task, Edge), ET , Rel_{req} , $deadline_{req}$
Output: Total EC

- 1: Turn on all VM's CPUs;
- 2: Perform RMST;
- 3: If Rel_{req} , $deadline_{req}$ are not met
- 4: Return false;
- 5: End if
- 6: While exists a turned-on VM's CPU do
- 7: For VM's CPU do
- 8: Perform RMST;
- 9: If Rel_{req} , $deadline_{req}$ are not met
- 10: Calculation of total EC;
- 11: End if
- 12: End for
- 13: If shutting down any VM's CPU would not met Rel_{req} , $deadline_{req}$
- 14: Break;
- 15: Else
- 16: Shut down the VM's CPU that can produces the minimum total EC;
- 17: End if
- 18: End while
- 19: Return the total EC;

3. Dynamic Voltage Frequency Scaling (DVFS)

Input: w (Task, Edge), ET , Rel_{req} , $deadline_{req}$
Output: The results of the workflow scheduling

- 1: For each $t_{o(j)} = t_{o(N1)} \rightarrow t_{o(1)}$ do
- 2: Calculate the $Rel_{req}^{o(j)}$;
- 3: For each $vm_k \in$ VMs;
- 4: Calculate $\Delta_{o(j),k}$;
- 5: Calculate $et_{o(j),k}'$;
- 6: Mapping the $t_{o(j)}$ and scale down VM's CPU frequency using Eq.(54);
- 7: Calculate the EC using Eq.(5);
- 8: End for
- 9: Finding the VM who has lower EC.
- 10: Mapping the $t_{o(j)}$ to this VM and scale down its VM's CPU frequency using Eq.(54);
- 11: End for

Algorithm 1. The proposed workflow scheduling method.

Evaluation metrics and experimental results

In this section, firstly, how to perform the simulation, the type of dataset, the features of the fog-cloud environment, the compared algorithms, the evaluation metrics used, and then the results of the experimental tests (based on the evaluation metrics) in order to evaluate the performance of the workflow scheduling method is described. In this paper, MATLAB R2018b software is used as a test environment to implement the proposed method and comparative methods on ASUS laptops with Intel Core i5 processor with a frequency of 2.50 GHz and 8 GB of memory with the operating system 64-bit Windows 7 is installed.

Four scientific workflows with different numbers of tasks and sizes (simple or complex tasks) are used to evaluate and compare algorithms (four synthetic workflows of Small (S), Medium (M), Large (L), and Extra-Large (EL) with various task number of 30, 60, 100, 1000, respectively). These workflows include Montage, Epigenomics, LIGO, and SPHIT⁸⁸. Therefore, different workflows require different storage capacities, CPU capacities, as well as different processing capacities when processing on VMs on fog/cloud computing^{4,65}. Figure 5 shows the general structure of the four scientific workflows used. Figure 5(a) shows the architecture of the Epigenomics workflow. Figure 5(b) shows the LIGO workflow structure. Also, Fig. 5(c) shows the SIPHT workflow structure. Figure 5(d) shows the overall structure of the Montage workflow⁸⁸. Montage workflow requires heavy CPU usage to process tasks that are mostly I/O sensitive. Epigenomics workflows are used in bioinformatics and are available to generate genome sequences automatically. LIGO workflow requires higher CPU and memory usage while processing tasks. SIPHT workflow structure is a program that uses a single workflow to automate the search for RNAs-encoding genes for all bacterial clones. It was developed for a bioinformatics project at Harvard University looking for untranslated RNAs to regulate processes such as virulence or secretion in bacteria^{4,41,65,89,90}. In all experiments, the fog-cloud environment consists of 3 data centers, 8 PMs (host), and 80 VMs with different configurations. The configuration details of PMs and VMs are presented in Table 3. As shown in Table 3, the slowest and fastest VM processing power is 100 and 5000 MIPS, respectively.

In the following, we will describe the evaluation metrics. Also, we describe the experimental results of each evaluation metric and the operation of comparing the proposed method with other methods based on each

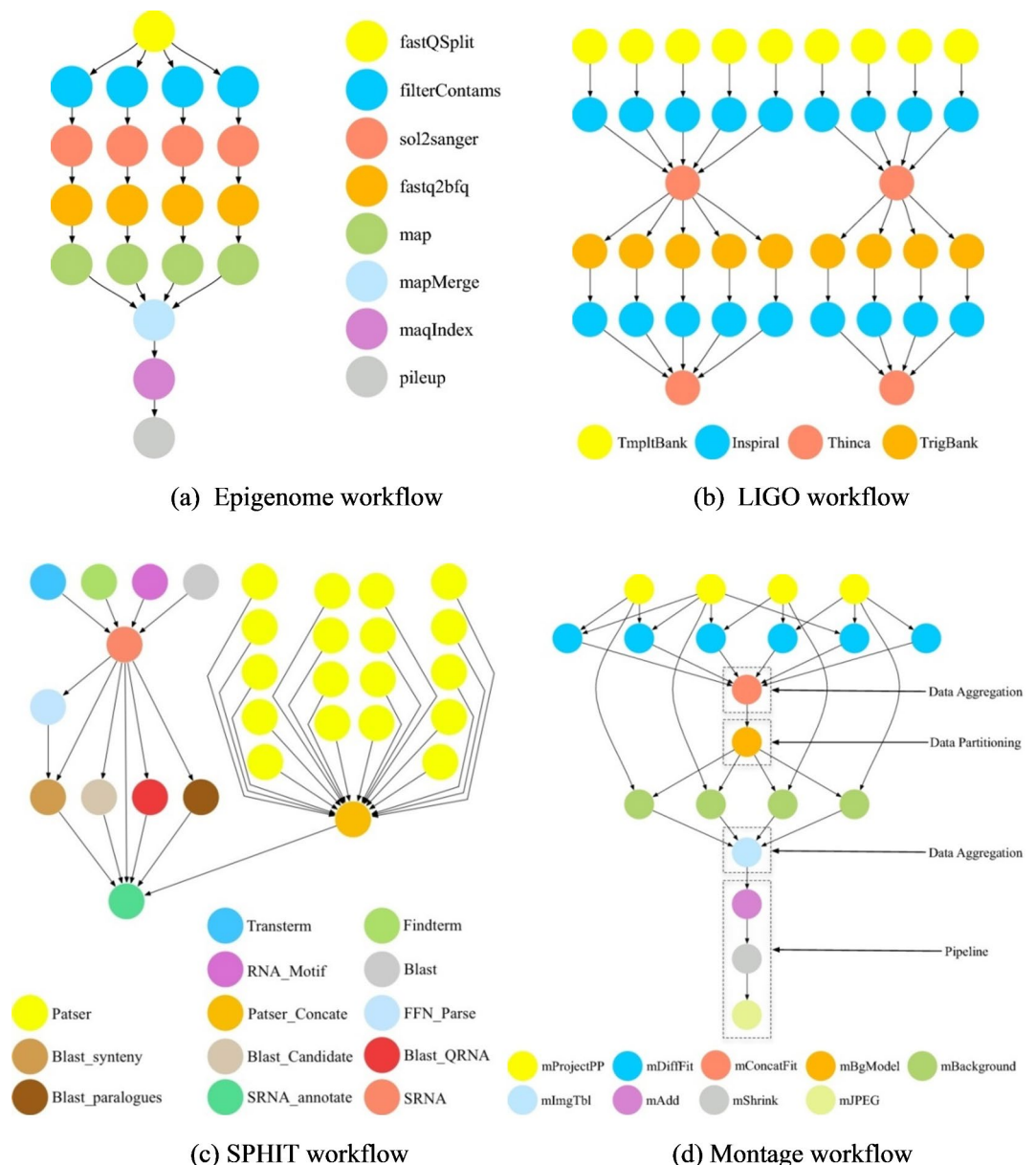


Fig. 5. Applied scientific workflows⁸⁸.

	Specification	Amount
Users	Users count	900
Datacenter	Data centers count	3
Physical meshine	Hosts count	8
	Storage	1 TB
	Network bandwidth	100 Mb/s
	RAM size	6 GB
Virtual machine	VMs count	80
	CPU capacity	[100, 5000] MIPS
	Storage	20 GB
	RAM size	1 GB
	Network bandwidth	100 Mb/s
	Processor	Xen
	Processors' count	1

Table 3. Experiment parameters.

	Specification	Amount
FA ⁷⁴	E ₀	[-1, 1]
	A	0.5
PSO ⁷³	c ₁	1.49
	c ₂	1.49
HHO ⁷⁵	b	0.2
	Γ	1
	W	0.9→0.4
DE-MFO ⁴	A	0.4
	β	0.3
	γ	0.3
AO ⁷²	A	0.1
	Δ	0.1
SSA ⁷¹	c ₁	[0,1]
	c ₂	[0,1]
	c ₃	[0,1]

Table 4. Parameters' settings in the proposed algorithm and compared algorithms.

evaluation metric. The experimental results are the result of 30 different runs of the optimization algorithms and the averaging of the 30 relevant runs. According to the MH algorithms used for task/workflow scheduling by previous researchers in the literature, the proposed ASSA with FA⁷⁴, AO⁷², HHO⁷⁵, PSO⁷³, and SSA⁷¹ are compared. Also, considering that the DVFS technique is used in the proposed method to reduce EC, the proposed ASSA is compared with DE-MFO⁴ and DVFS⁴¹ methods which are based on the DVFS technique. Table 4 shows the fixed parameters specific to each MH algorithm used for comparison. All comparisons are in the same conditions (the same number of iterations (100 iterations), the same stopping condition (reaching the 100-th iteration), the same number of runs (30 runs), the same objective function (Eq. (15)), the same number of search agents (50 search agents), the same dataset.

Evaluation metrics

In this section, the evaluation metrics examined are described. The reliability ratio of a workflow w is defined by the Eq. (56):

$$Relratio = \frac{requiredreliability}{maximumreliability} = \frac{Rel_{req}(w)}{Rel_{max}(w)} \quad (56)$$

Also, for deadlines ratio:

$$Deadlineratio = \frac{workflowdeadline}{schedulingfromRMT} = \frac{deadline(w)}{scheduling_{RMT}(w)} \quad (57)$$

In order to evaluate the performance of the proposed workflow scheduling method, according to^{41,91–94}, $f_{k,max}$ = 1.2, $0.9 \leq \alpha \leq 1.3$, $2.5 \leq m_j \leq 3.1$ are considered. The CPU frequency of all VMs is increased with an accuracy of 0.01 GHz⁹⁰.

Energy consumption

In this section, we perform our experiments for variable deadline ratio and variable reliability ratio, different workflows with variable number of tasks, and examine the EC results.

In all workflows used, the amount of data that is exchanged between tasks of the workflow is in kilojoule, so the amount of EC is also in kilojoule. In Table 5, the results of EC for different workflows with different numbers of tasks are shown by different algorithms with deadline ratio = 1.6 and reliability ratio = 0.92. According to Table 5, as the number of tasks in workflows increases, the EC for all algorithms and methods increases. Because as the number of tasks in each workflow increases, the number of VMs used also increases. If the number of VMs used in the scheduling increases, it causes the CPU of these VMs to stay on, and the EC of the fog-cloud system increases. Also, with the increase in the number of tasks in a workflow, it takes a lot of time for the tasks to be executed and completed on the selected VMs, which again causes the CPU of the VMs to stay on for a long time. According to Table 5, our proposed method, which uses the combination of ASSA with RMST, VM merging (the EC performance of VM merging is not affected by the number of VMs), and DVFS techniques, is always more efficient than methods FA, AO, HHO, PSO, SSA, DE-MFO⁴ and DVFS⁴¹. Because in our proposed method, the CPU of unnecessary VMs is shut down by VM merging the static EC is reduced, and the CPU frequency of VMs is adjusted by DVFS when executing tasks. The DVFS technique⁴¹ at the task level reduces the task execution frequency of the workflow and changes the VM processing frequency when performing different tasks. In the DMFO-DE method⁴, the MST and EC metrics are considered, and in it, EC is controlled using the DVFS technique. As can be seen from Table 5, for all workflows with different numbers of tasks, ASSA consumed the least amount of energy, followed by AO, HHO, SSA, FA, PSO, DE-MFO⁴ and DVFS⁴¹, respectively. Then AO, HHO, SSA, FA, PSO, DE-MFO⁴, and DVFS⁴¹ consumed the least amount of energy, respectively.

Figure 6 shows the difference in EC values of ASSA, AO, HHO, PSO, SSA, FA, and DE-MFO⁴ compared to DVFS⁴¹ for deadline ratio = 1.6 and reliability ratio = 0.92. According to the diagrams in Fig. 6, ASSA has the most difference with DVFS⁴¹. Also, the proposed ASSA has a significant difference in the EC value with AO, HHO, SSA, PSO, FA, and DE-MFO⁴, and it consumes less energy than AO, HHO, SSA, PSO, FA, DVFS⁴¹, and DE-MFO⁴ methods.

In Table 6, the results of EC for different workflows with different numbers of tasks are shown by different algorithms for various reliability ratios and a deadline ratio of 1.6 (For workflows with the number of tasks 30 (S) and 60 (EL)). According to Table 6, as the number of tasks in workflows increases, the EC for all algorithms and methods increases. With a fixed deadline ratio of 1.6 and an increase in reliability ratio, EC increases for all workflows. The higher the reliability of the system, the more possibility the tasks will be executed and completed, and this will cause more VMs to be active for a longer period. Keeping the VM processor on increases EC.

Figure 7 shows the difference in EC values of ASSA, AO, HHO, SSA, PSO, FA, and DE-MFO⁴ compared to DVFS⁴¹ for deadline ratio = 1.6 and various reliability ratios. According to the diagrams in Fig. 7, ASSA has the most difference with DVFS⁴¹. Also, the proposed ASSA has a significant difference in the EC value with AO, HHO, SSA, PSO, FA, and DE-MFO⁴, and it consumes less energy than AO, HHO, SSA, PSO, FA, DVFS⁴¹, and DE-MFO⁴ methods.

Algorithm	Montage	Epigenomics	LIGO	SIPTH	Montage	Epigenomics	LIGO	SIPTH
	S workflow				M workflow			
ASSA	1207	1300	1252	1350	1700	1850	1920	1880
AO	1240	1330	1290	1370	1735	1883	1948	1906
PSO	1380	1460	1430	1470	1800	1980	2030	2010
HHO	1270	1370	1331	1402	1750	1900	1960	1935
FA	1350	1431	1390	1430	1782	1952	2003	1983
SSA	1305	1390	1360	1440	1770	1930	1980	1952
DE-MFO ⁴	1422	1505	1480	1503	1831	2005	2051	2035
DVFS ⁴¹	1457	1540	1511	1544	1853	2036	2073	2070
	L workflow				EL workflow			
ASSA	2320	2496	2520	2580	10,000	11,500	11,100	11,800
AO	2338	2520	2550	2600	10,121	11,620	11,210	11,915
PSO	2430	3010	2650	2096	10,490	12,104	11,650	12,360
HHO	2363	2550	2578	2627	10,210	11,710	11,325	12,035
FA	2401	2594	2626	2074	10,400	11,920	11,547	12,250
SSA	2380	2576	2600	2050	10,305	11,802	11,430	12,130
DE-MFO ⁴	2455	3028	2682	3021	10,580	12,230	11,765	12,465
DVFS ⁴¹	2480	3051	2905	3043	10,695	12,310	11,870	12,580

Table 5. Mean EC results from algorithms for different datasets (30 runs) for deadline ratio = 1.6 and reliability ratio = 0.92. Significant values are in [bold].

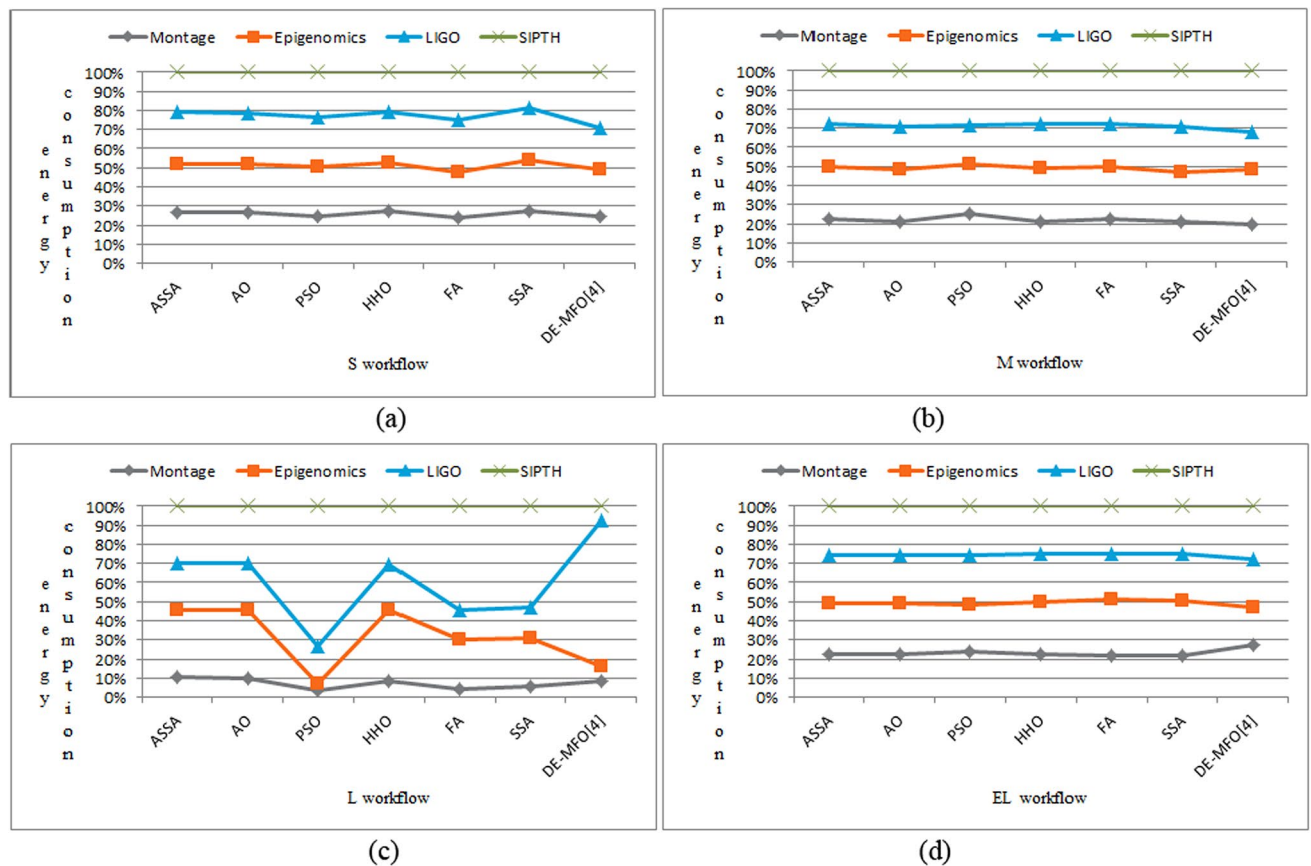


Fig. 6. Difference in EC values of ASSA, AO, HHO, SSA, PSO, FA, and DE-MFO⁴ compared to DVFS⁴¹ for deadline ratio = 1.6 and reliability ratio = 0.92.

Table 7 shows the system EC for deadline ratio $\in [1.4, 2]$ and fixed reliability ratio = 0.92. By increasing the deadline ratio, EC is also reduced for both small and large workflows. The lower the deadline ratio is, the fog-cloud system suffers from a bottleneck where a number of tasks must be executed by a shorter deadline. But when the deadline ratio increases, it is the reliability requirements that become the main bottleneck in the fog-cloud system, which causes less energy to be consumed in the fog-cloud system. The ASSA has consumed less energy than DVFS⁴¹ and other methods.

Figure 8 shows the difference in EC values of ASSA, AO, HHO, SSA, PSO, FA, and DE-MFO⁴ compared to DVFS⁴¹ for different deadline ratios and fixed reliability ratios. According to the diagrams in Fig. 8, ASSA has the most difference with DVFS⁴¹.

Also, the proposed ASSA has a significant difference in the EC value with AO, HHO, SSA, PSO, FA, and DE-MFO⁴, and it consumes less energy than AO, HHO, SSA, PSO, FA, DVFS⁴¹, and DE-MFO⁴ methods.

MakeSpan Time

Table 8 shows the value of the MST obtained by ASSA and the compared algorithms. As can be seen from Table 8, for all workflows with different sizes, the MST value obtained by ASSA is lower than the MST value obtained by FA, AO, HHO, PSO, DE-MFO, SSA. Then, AO, HHO, SSA, DE-MFO, FA, and PSO have obtained the lowest value of the MST, respectively.

Figure 9 shows the difference in MST values of ASSA, AO, HHO, SSA, PSO, FA, and DE-MFO⁴ compared to DVFS⁴¹ for deadline ratio = 1.6 and reliability ratio = 0.92. According to the diagrams in Fig. 9, ASSA has the most difference with DVFS⁴¹. Also, the proposed ASSA has a significant difference in the MST value with AO, HHO, SSA, PSO, FA, and DE-MFO⁴, and it has less MST than AO, HHO, SSA, PSO, FA, DVFS⁴¹, and DE-MFO⁴ methods.

Objective function value

In all problems that MH algorithms are used to solve, the objective function is very important and its design plays an important role in determining the final solutions of the problem as well as the degree of convergence of the MH algorithm. In problems that use the minimization of the objective function, the MH algorithm obtains a lower value of the objective function in each iteration than the previous iteration, and by minimizing the objective function, it obtains optimal solutions. So, suppose the value of the objective function obtained for an algorithm is lower than the value of the objective function obtained by the compared algorithms. In that

Reliability ratio	0.92	0.94	0.96	0.98	0.92	0.94	0.96	0.98
	S workflow(Montage)				EL workflow(Montage)			
ASSA	1207	1320	1410	1580	10,000	10,200	10,400	10,600
AO	1240	1392	1480	1660	10,121	10,300	10,510	10,720
PSO	1380	1587	1693	2050	10,490	10,750	10,875	11,160
HHO	1270	1450	1540	1780	10,210	10,410	10,605	10,800
FA	1350	1535	1637	1970	10,400	10,640	10,780	11,070
SSA	1305	1496	1595	1865	10,305	10,525	10,695	10,910
DE-MFO ⁴	1422	1640	1764	2125	10,580	10,835	10,980	11,265
DVFS ⁴¹	1457	1698	1830	2205	10,695	10,940	11,100	11,364
	S workflow(Epigenomics)				EL workflow(Epigenomics)			
ASSA	1300	1440	1560	1650	11,500	10,700	10,900	11,050
AO	1330	1535	1680	1700	11,620	10,800	11,050	11,145
PSO	1460	1943	2020	2076	12,104	11,275	11,450	11,589
HHO	1370	1612	1770	1798	11,710	10,920	11,130	11,260
FA	1431	1800	1910	1985	11,920	11,180	11,355	11,464
SSA	1390	1700	1860	1890	11,802	11,090	11,240	11,358
DE-MFO ⁴	1505	2005	2130	2180	12,230	11,370	11,560	11,680
DVFS ⁴¹	1540	2090	2215	2295	12,310	11,485	11,658	11,783
	S workflow(LIGO)				EL workflow(LIGO)			
ASSA	1252	1380	1500	1610	11,100	11,200	11,350	11,550
AO	1290	1490	1590	1700	11,210	11,305	11,448	11,679
PSO	1430	1848	1960	2158	11,650	11,700	11,860	12,155
HHO	1331	1590	1680	1815	11,325	11,400	11,545	11,790
FA	1390	1760	1850	2070	11,547	11,601	11,753	12,020
SSA	1360	1650	1775	1910	11,430	11,497	11,650	11,900
DE-MFO ⁴	1480	1950	2078	2255	11,765	11,810	11,957	12,270
DVFS ⁴¹	1511	2055	2180	2360	11,870	11,908	12,050	12,387
	S workflow(SIPTH)				EL workflow(SIPTH)			
ASSA	1350	1490	1590	1700	11,800	11,000	11,200	11,350
AO	1370	1595	1712	1802	11,915	11,110	11,310	11,452
PSO	1470	1913	2102	2201	12,360	11,594	11,718	11,870
HHO	1402	1700	1797	1913	12,035	11,246	11,405	11,560
FA	1430	1805	2017	2110	12,250	11,485	11,620	11,773
SSA	1440	1790	1906	2008	12,130	11,363	11,512	11,675
DE-MFO ⁴	1503	2000	2200	2223	12,465	11,700	11,823	11,986
DVFS ⁴¹	1544	2111	2307	2318	12,580	11,812	11,925	12,050

Table 6. Mean EC results from algorithms for different datasets (30 runs) for deadline ratio 1.6 Significant values are in [bold].

case, it can be relatively said that the corresponding algorithm has been able to obtain better solutions than the compared algorithms.

Figure 10 shows the value of the objective function obtained by ASSA and the compared algorithms with deadline ratio = 1.6 and reliability ratio = 0.92. The value of the objective function is calculated using Eq. (15). As can be seen from Fig. 10, for all workflows with different sizes, the objective function value obtained by ASSA is lower than the objective function value obtained by FA, AO, HHO, PSO, DE-MFO, SSA. Then, AO, HHO, SSA, DE-MFO, FA, and PSO obtained the lowest value of the objective function, respectively.

Figure 11, 12, 13, and 14 indicates the investigated algorithms' investigated objective function values applied to schedule Montage, LIGO, Epigenomics, and SIPTH workflows with 30 tasks (S workflow), 60 tasks (M workflow), 100 tasks (L workflow), 1000 tasks (EL workflow), respectively. Algorithms have been repeated 100 times and all algorithms have reached convergence in almost 70-th iterations. As shown in these figures, ASSA can rapidly converge to the near-optimal solution and achieve better results than other algorithms. Also, at first, the value of the objective function is the same for all solutions in all algorithms because the schedules obtained from the random solutions created by the corresponding algorithms cannot meet the set deadline and obtain the minimum EC value. But, after running the algorithms and increasing the number of iterations, the optimization algorithms can find better schedulings and reduce the objective function value, for which our method provides better results.

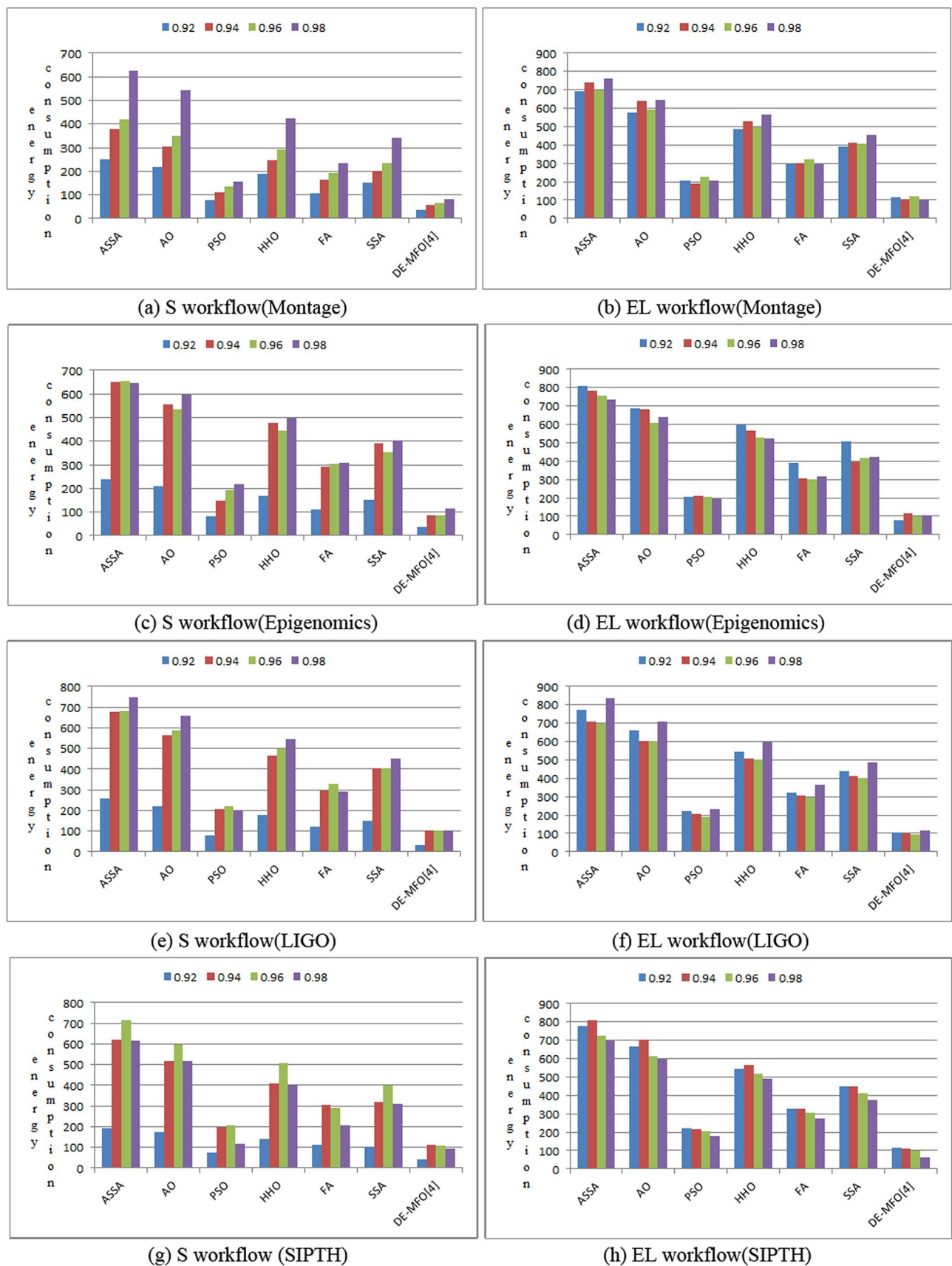


Fig. 7. Difference in EC values of ASSA, AO, HHO, SSA, PSO, FA, and DE-MFO⁴ compared to DVFS⁴¹ for deadline ratio = 1.6 and various reliability ratio $\in [0.92 \text{ to } 0.98]$.

Deadline Ratio	1.4	1.6	1.8	2	1.4	1.6	1.8	2
	S workflow(Montage)				EL workflow(Montage)			
ASSA	1207	1322	1410	1580	10,000	10,200	10,410	10,600
AO	1240	1445	1523	1660	10,121	10,312	10,529	10,723
PSO	1380	1894	1943	2100	10,490	10,724	10,846	11,155
HHO	1270	1550	1618	1780	10,210	10,425	10,630	10,845
FA	1352	1760	1832	1991	10,400	10,632	10,740	11,050
SSA	1305	1643	1724	1875	10,305	10,520	10,732	10,944
DE-MFO ⁴	1422	2100	2045	2211	10,580	10,834	10,950	11,249
DVFS ⁴¹	1457	2195	2155	2320	10,695	10,940	11,063	11,354
	S workflow(Epigenomics)				EL workflow(Epigenomics)			
ASSA	1300	1440	1560	1650	11,500	10,700	10,900	11,050
AO	1330	1538	1654	1767	11,620	10,812	11,017	11,145
PSO	1460	1952	2018	2140	12,104	11,222	11,425	11,570
HHO	1370	1642	1750	1870	11,710	10,910	11,115	11,252
FA	1431	1840	1900	2033	11,920	11,120	11,320	11,455
SSA	1390	1745	1862	1991	11,802	11,018	11,224	11,349
DE-MFO ⁴	1505	2060	2100	2251	12,230	11,319	11,538	11,665
DVFS ⁴¹	1540	2116	2205	2349	12,310	11,430	11,640	11,774
	S workflow(LIGO)				EL workflow(LIGO)			
ASSA	1252	1380	1500	1610	11,100	11,200	11,350	11,550
AO	1290	1495	1599	1700	11,210	11,328	11,472	11,672
PSO	1430	1886	2060	2179	11,650	11,780	11,900	12,080
HHO	1331	1589	1711	1820	11,325	11,452	11,571	11,770
FA	1390	1787	1952	2070	11,547	11,645	11,790	11,985
SSA	1360	1691	1835	1960	11,430	11,549	11,682	11,868
DE-MFO ⁴	1480	1998	2189	2310	11,765	11,898	12,020	12,200
DVFS ⁴¹	1511	2112	2300	2425	11,870	12,000	12,153	12,324
	S workflow(SIPTH)				EL workflow(SIPTH)			
ASSA	1350	1490	1598	1700	11,800	11,000	11,200	11,350
AO	1370	1610	1712	1843	11,915	11,146	11,321	11,452
PSO	1470	2050	2187	2296	12,360	11,592	11,745	11,860
HHO	1402	1701	1837	1976	12,035	11,265	11,418	11,551
FA	1430	1940	2068	2170	12,250	11,480	11,642	11,766
SSA	1440	1820	1949	2089	12,130	11,387	11,528	11,662
DE-MFO ⁴	1503	2155	2285	2411	12,465	11,690	11,849	11,976
DVFS ⁴¹	1544	2298	2413	2520	12,580	11,794	11,962	12,127

Table 7. Mean EC results from algorithms for different datasets (30 runs) for Reliability Ratio = 0.92.

Conclusion

In a fog-cloud computing system, providing a scheduling algorithm and an efficient architecture to reduce EC for workflow scheduling is important. This paper proposed RMST, DVFS, and VM merging techniques, as well as the combined MH algorithm of ASSA, in order to reduce EC and workflow execution time while maintaining deadline, reliability, and Real-Time constraints. ASSA was repeated 100 times. In ASSA, at the beginning of the optimization, a number of VMs were randomly selected by ASSA. Then, while maintaining the requirements of reliability and deadline, the RMST technique was used to reduce the MST. While maintaining the requirements of the MST, reliability, and deadline, the VM merging technique was used to reduce the static EC of the selected VMs, and DVFS was used to reduce dynamic EC. By using RMST, the MST to complete and execute workflows was reduced, and by using VM merging and DVFS, the total EC when executing workflows was minimized. Finally, by using the minimization of the objective function, the most optimal VMs with the lowest MST, the highest level of reliability, and the lowest EC were obtained. The experimental results show that compared to the FA⁷⁴, AO⁷², HHO⁷⁵, PSO⁷³, SSA⁷¹, DE-MFO⁴, and DVFS⁴¹ methods, the proposed ASSA has been able to perform better in terms of EC and MST. The reason for the superiority of the proposed method over the compared methods is that we used the combination of several techniques, Reducing MakeSpan Time (RMST), VM merging, and Dynamic Voltage Frequency Scaling (DVFS) in a stepwise manner to improve energy and delay. Also, we used the combined metaheuristic algorithm to optimize our problem. The combination of all these techniques and the combined algorithm made our work better than the comparative works. Our future work is to improve this algorithm and system in terms of security and create a secure schedule that is very important in our world.

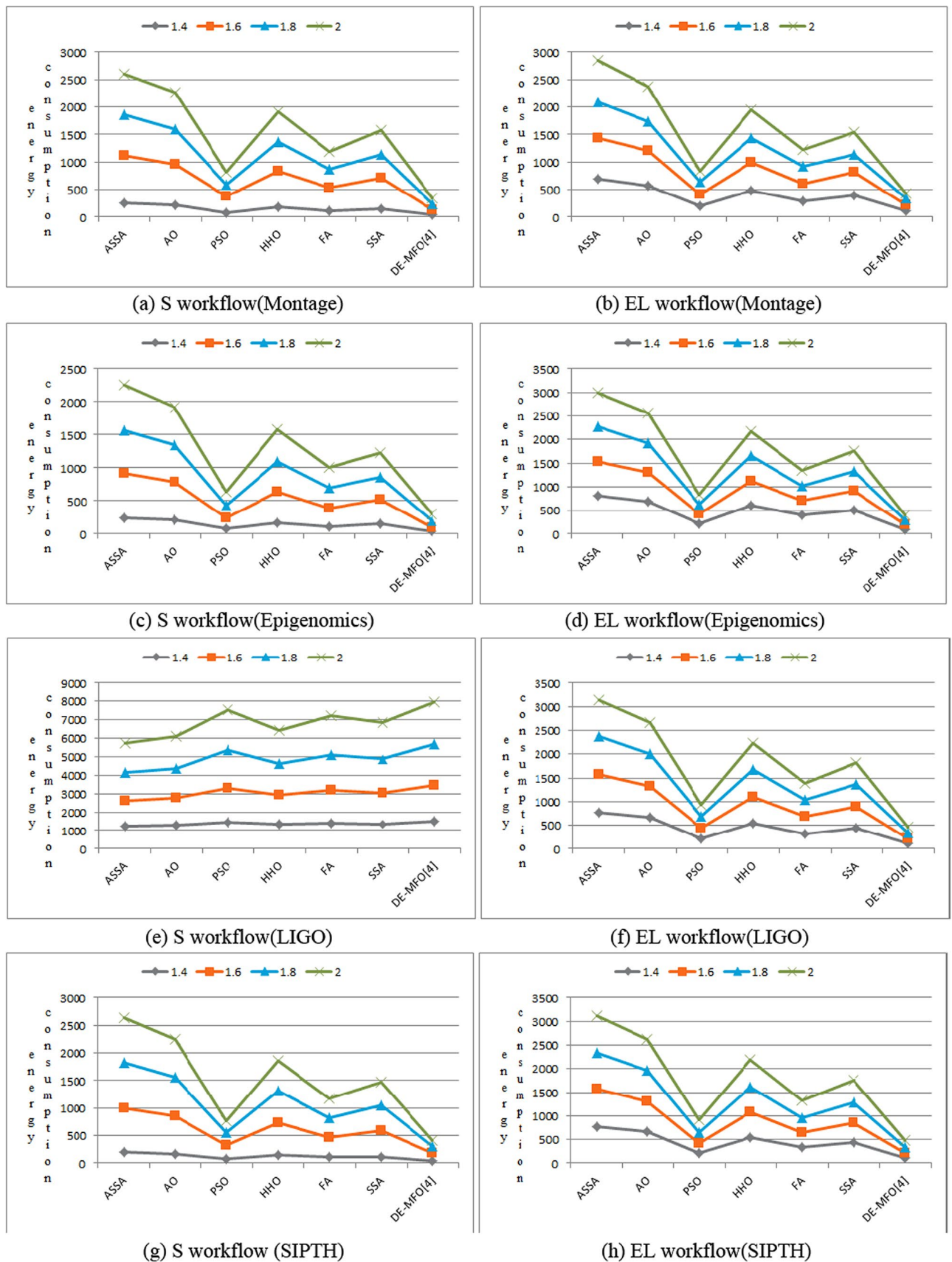


Fig. 8. Difference in EC values of ASSA, AO, HHO, SSA, PSO, FA, and DE-MFO⁴¹ compared to DVFS⁴¹ for deadline ratio $\in [1.4, 2]$ and fixed reliability ratio = 0.92.

Algorithm	Montage	Epigenomics	LIGO	SIPTH	Montage	Epigenomics	LIGO	SIPTH
	S workflow				M workflow			
ASSA	221	475	485	379	337	491	474	485
PSO	324	452	575	462	441	569	585	586
SSA	259	412	520	427	369	523	531	534
FA	302	445	562	455	436	559	563	574
HHO	245	403	508	411	354	513	517	525
AO	233	487	492	396	345	505	495	499
DE-MFO ⁴	271	424	531	433	373	535	540	556
DVFS ⁴¹	285	431	549	448	397	548	558	567
	L workflow				EL workflow			
ASSA	403	580	601	591	720	806	819	779
PSO	537	686	698	678	724	891	895	883
SSA	458	628	643	627	759	842	852	822
FA	515	674	681	666	735	879	883	864
HHO	434	611	631	614	746	834	841	810
AO	425	597	616	602	731	817	830	796
DE-MFO ⁴	464	639	652	645	773	854	866	835
DVFS ⁴¹	476	654	669	657	785	864	875	858

Table 8. Mean MST results for Different Datasets (30 runs) for deadline ratio = 1.6 reliability ratio = 0.92.

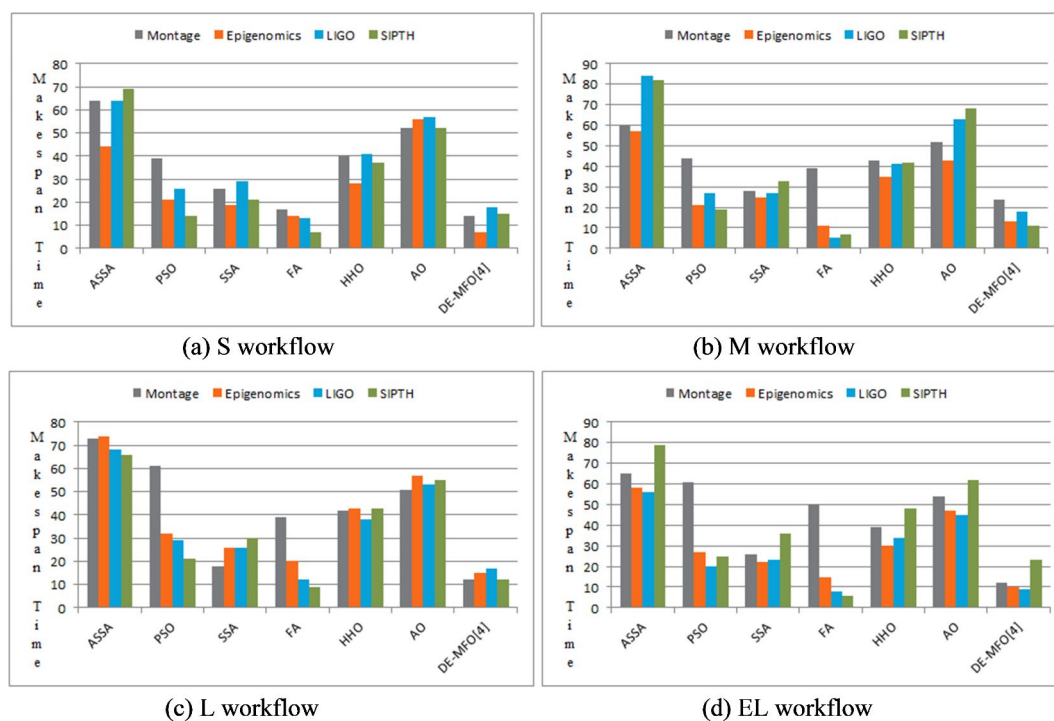


Fig. 9. Difference in MST values of ASSA, AO, HHO, PSO, SSA, FA, and DE-MFO⁴ compared to DVFS⁴¹ for deadline ratio = 1.6 reliability ratio = 0.92.

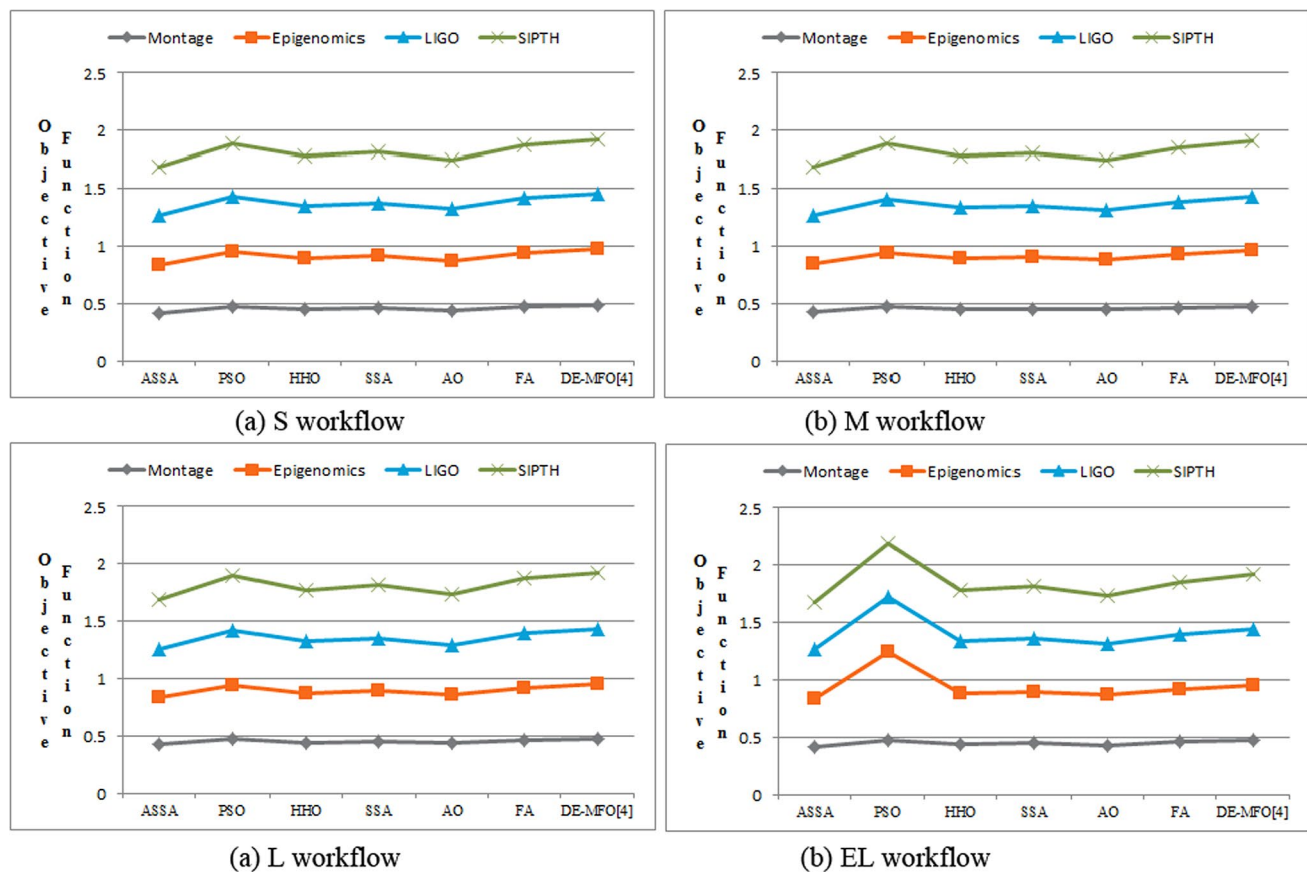
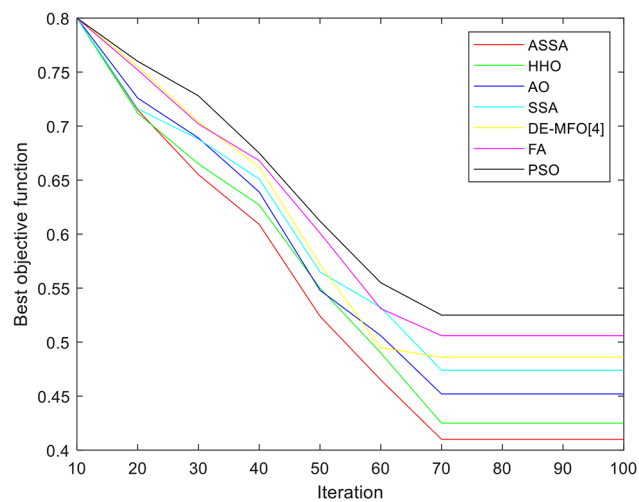
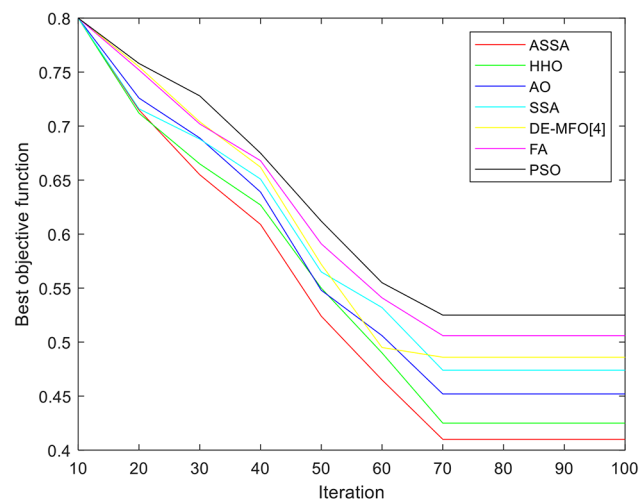


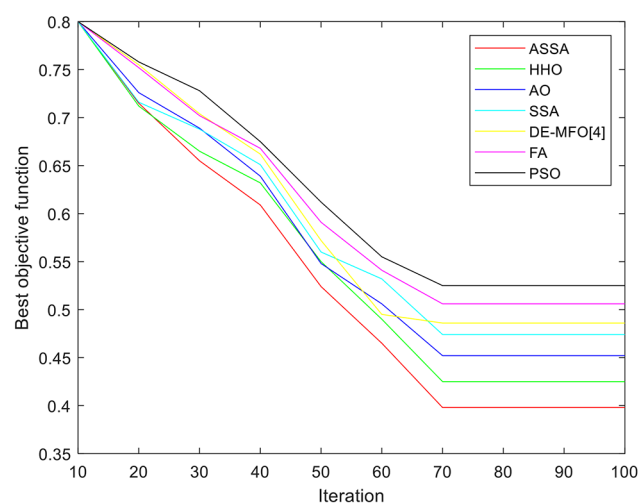
Fig. 10. Objective function values for different workflows with deadline ratio = 1.6 and reliability ratio = 0.92.



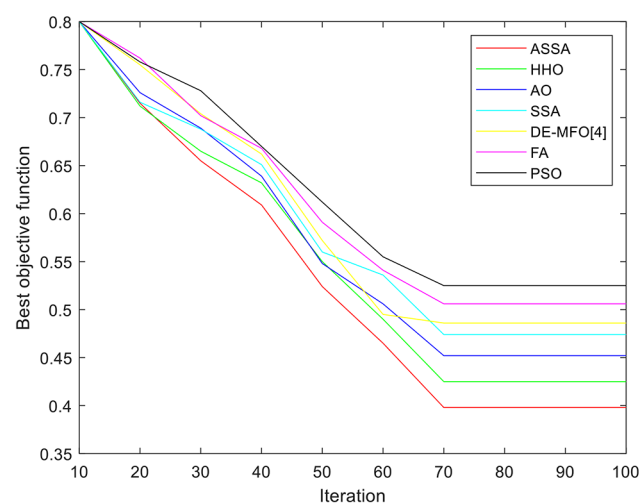
(a) Montage workflow



(b) Epigenomics workflow

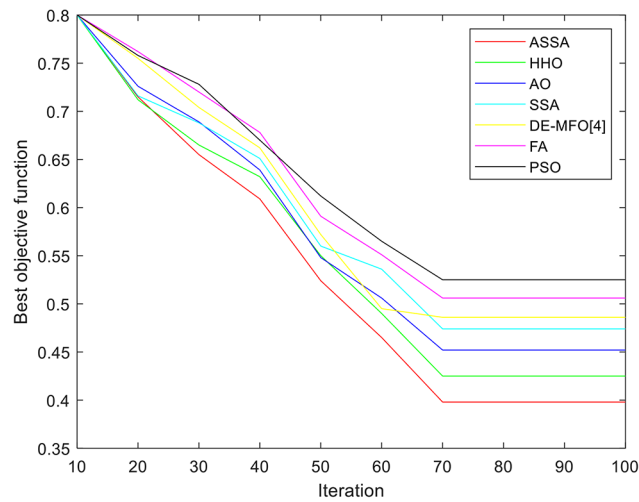


(c) LIGO workflow

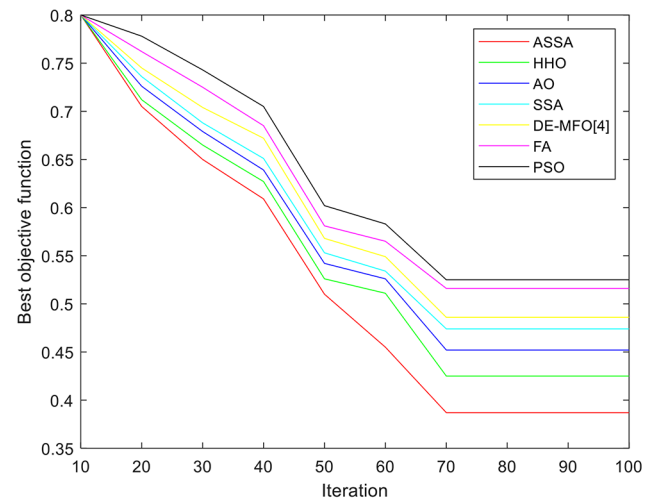


(d) SIPTH workflow

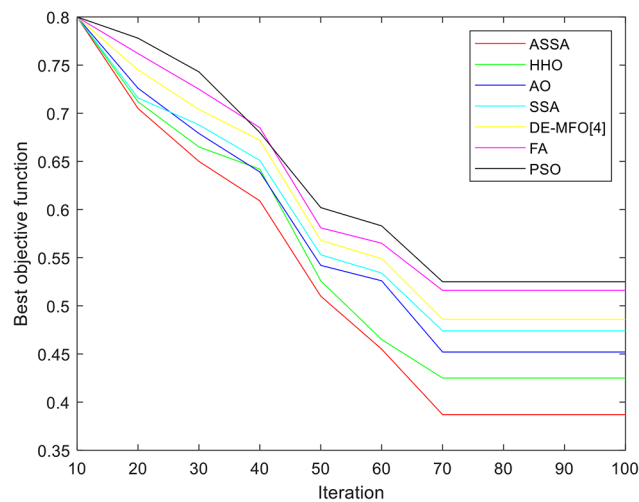
Fig. 11. Objective function values for S workflows.



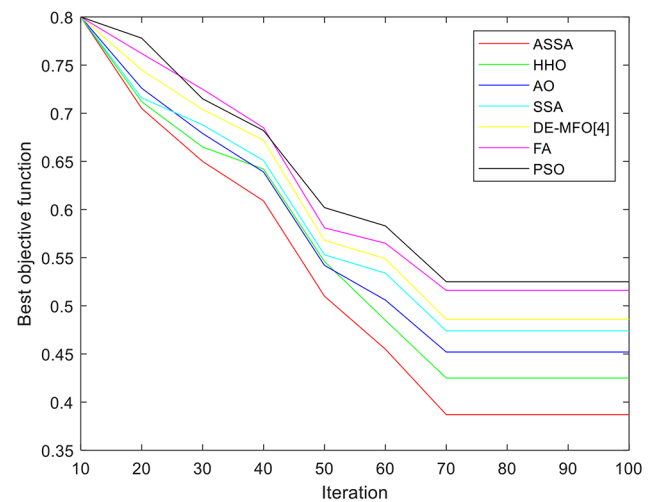
(a) Montage workflow



(b) Epigenomics workflow

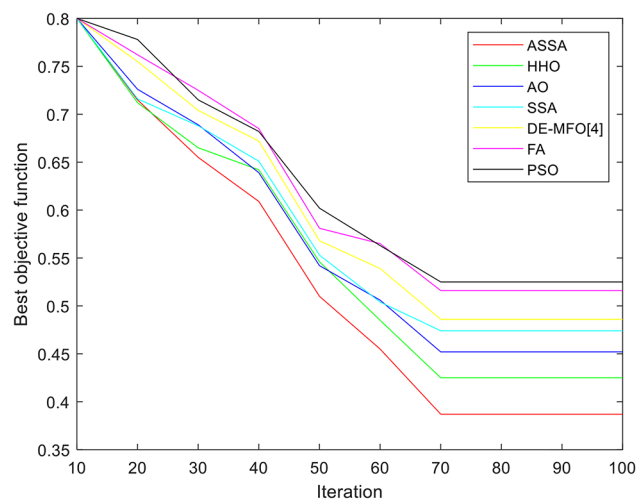


(c) LIGO workflow

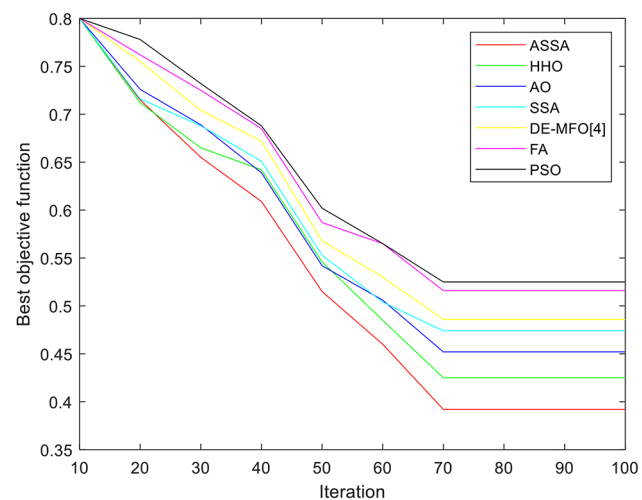


(d) SIPTH workflow

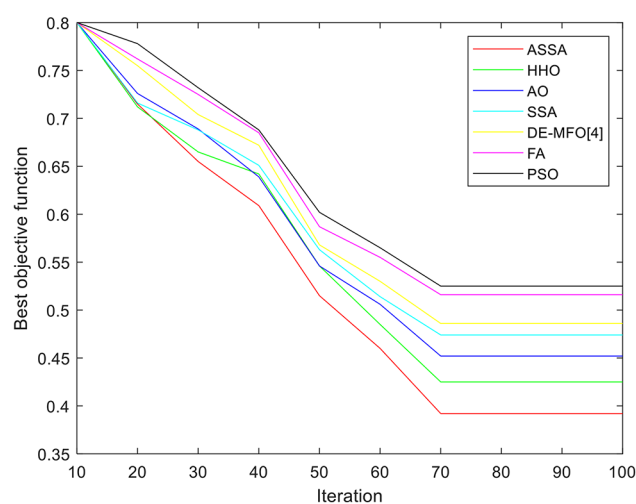
Fig. 12. Objective function values for M workflows.



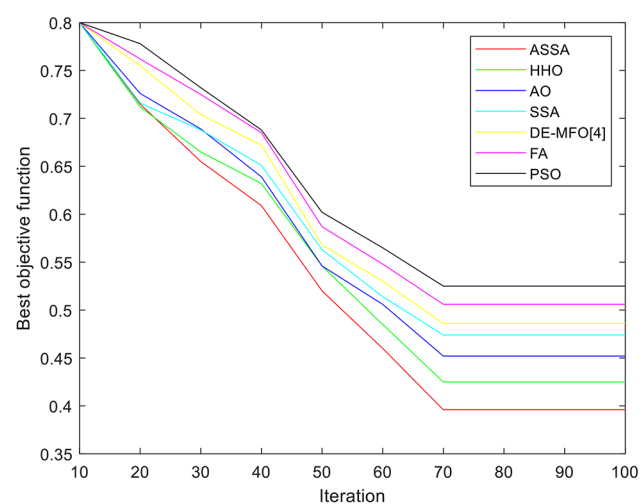
(a) Montage workflow



(b) Epigenomics workflow

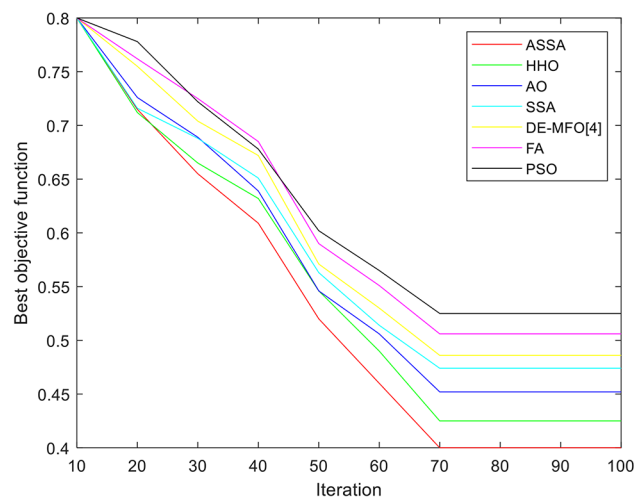


(c) LIGO workflow

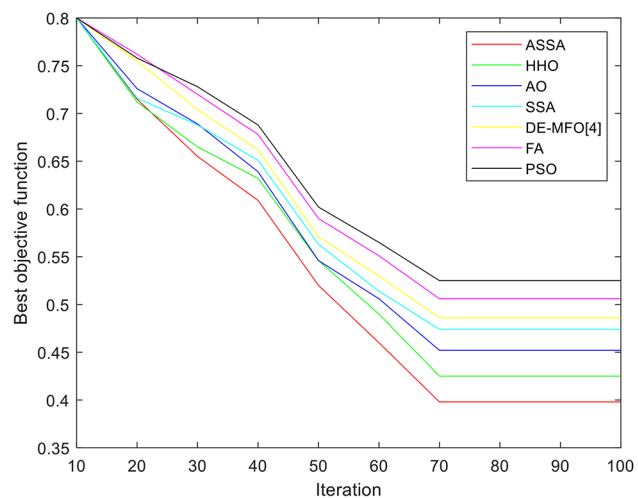


(d) SIPTH workflow

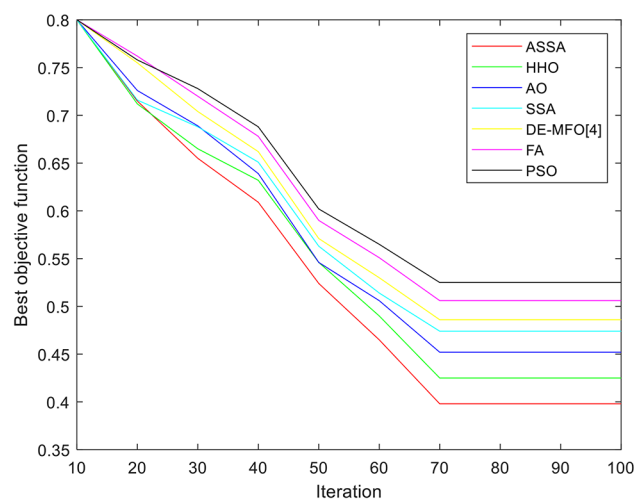
Fig. 13. Objective function values for L workflows.



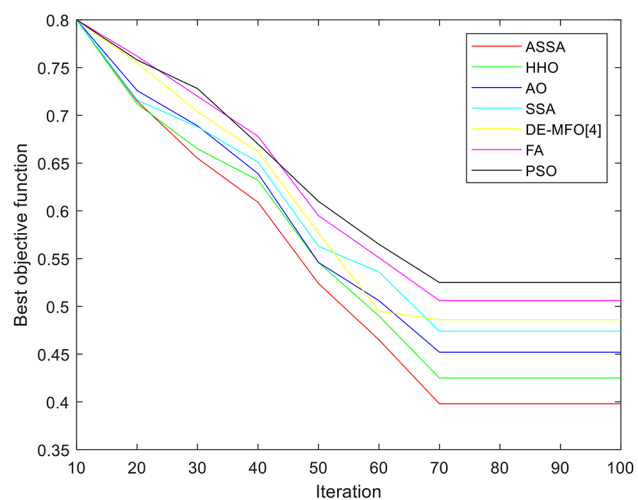
(a) Montage workflow



(b) Epigenomics workflow



(c) LIGO workflow



(d) SIPTH workflow

Fig. 14. Objective function values for EL workflows.

Data availability

Data is available from the Corresponding Author upon reasonable request.

Received: 9 July 2024; Accepted: 14 January 2025

Published online: 29 January 2025

References

- Masdari, M. et al. Analysis of secure LEACH - based clustering protocols in wireless sensor networks. *J. Netw. Comput. Appl.* **36**, 1243–1260 (2013).
- Masdari, M. et al. CDABC: chaotic discrete artificial bee colony algorithm for multi-level clustering in largescale WSNs. *J. Supercomput.* **75**, 7174–7208 (2019).
- Masdari, M., & Khoshnevis, A. A survey and classification of the workload forecasting methods in cloud computing. *Clust. Comput.* 1–26 (2019).
- Ahmed, O. H. et al. Using differential evolution and Moth-Flame optimization for scientific workflow scheduling in fog computing. *Appl. Soft Comput.* **112**, 107744 (2021).
- Wortmann, F. & Flüchter, K. Internet of things. *Bus. Inf. Syst. Eng.* **57**, 221–224 (2015).
- Sun, G. et al. Low-latency and resource-efficient service function chaining orchestration in network function virtualization. *IEEE Internet Things J.* **7**(7), 5760–5772. <https://doi.org/10.1109/JIOT.2019.2937110> (2020).
- Sun, G. et al. Cost-efficient service function chain orchestration for low-latency applications in NFV networks. *IEEE Syst. J.* **13**(4), 3877–3888. <https://doi.org/10.1109/JSYST.2018.2879883> (2019).
- Shahidinejad, A. et al. Resource provisioning using workload clustering in cloud computing environment: A hybrid approach. *Clust. Comput.* **24**, 319–342 (2021).
- Sun, G., Liao, D., Zhao, D., Xu, Z. & Yu, H. Live migration for multiple correlated virtual machines in cloud-based data centers. *IEEE Trans. Serv. Comput.* **11**(2), 279–291. <https://doi.org/10.1109/TSC.2015.2477825> (2018).
- Sun, G., Xu, Z., Yu, H. & Chang, V. Dynamic network function provisioning to enable network in box for industrial applications. *IEEE Trans. Ind. Inform.* **17**(10), 7155–7164. <https://doi.org/10.1109/TII.2020.3042872> (2021).
- Sun, G., Wang, Z., Su, H., Yu, H., Lei, B. & Guizani, M. Profit maximization of independent task offloading in MEC-enabled 5G internet of vehicles. *IEEE Trans. Intell. Transp. Syst.* 1–13. <https://doi.org/10.1109/TITS.2024.3416300> (2024).
- Wang, R. et al. FI-NPI: Exploring optimal control in parallel platform systems. *Electronics* **13**(7), 1168. <https://doi.org/10.3390/electronics13071168> (2024).
- Bonomi, F. et al. Fog computing and its role in the internet of things. in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 13–16 (2012).
- Shakarami, A. et al. Data replication schemes in cloud computing: A survey. *Clust. Comput.* 1–35 (2021).
- Vaquero, L. M. & Roderio Merino, L. Finding your way in the fog: Towards a comprehensive definition of fog computing. *ACM SIGCOMM Comput. Commun. Rev.* **44**, 27–32 (2014).
- Hossain, M. R. et al. A scheduling-based dynamic fog computing framework for augmenting resource utilization. *Simul. Model. Pract. Theory*. 102336 (2021).
- Wang, Y. et al. Wireless multiferroic memristor with coupled giant impedance and artificial synapse application. *Adv. Electron. Mater.* **8**(10), 2200370. <https://doi.org/10.1002/aelm.202200370> (2022).
- Shahidinejad, A. et al. Context-aware multi-user offloading in mobile edge computing: A federated learning-based approach. *J. Grid Comput.* **19**, 1–23 (2021).
- Elaziz, M. A. et al. Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments. *Futur. Gener. Comput. Syst.* **124**, 142–154 (2021).
- Zheng, W. et al. Design of a modified transformer architecture based on relative position coding. *Int. J. Comput. Intell. Syst.* **16**(1), 168. <https://doi.org/10.1007/s44196-023-00345-z> (2023).
- Gong, Q., Li, J., Jiang, Z. & Wang, Y. A hierarchical integration scheduling method for flexible job shop with green lot splitting. *Eng. Appl. Artif. Intell.* **129**, 107595. <https://doi.org/10.1016/j.engappai.2023.107595> (2024).
- Cao, B., Zhao, J., Liu, X. & Li, Y. Adaptive 5G-and-beyond network-enabled interpretable federated learning enhanced by neuroevolution. *Sci China Inf Sci* **67**(7), 170306. <https://doi.org/10.1007/s11432-023-4011-4> (2024).
- Montazerolghaem, A. A. Efficient resource allocation for multimedia streaming in software-defined internet of vehicles. *IEEE Trans. Intell. Transp. Syst.* **24**(12), 14718–14731. <https://doi.org/10.1109/TITS.2023.3303404> (2023).
- Yin, L. et al. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. *IEEE Trans. Ind. Inform.* **14**, 4712–4721 (2018).
- Xu, X. & Wei, Z. Dynamic pickup and delivery problem with transshipments and LIFO constraints. *Comput. Ind. Eng.* **175**, 108835. <https://doi.org/10.1016/j.cie.2022.108835> (2023).
- Zhu, C. Intelligent robot path planning and navigation based on reinforcement learning and adaptive control. *J. Logist. Inform. Serv. Sci.* **10**(3), 235–248. <https://doi.org/10.33168/JLISS.2023.0318> (2023).
- Lu, J. & Osorio, C. Link transmission model: A formulation with enhanced compute time for large-scale network optimization. *Transp. Res. Part B Methodol.* **185**, 102971. <https://doi.org/10.1016/j.trb.2024.102971> (2024).
- Montazerolghaem, A. et al. Green cloud multimedia networking: NFV/SDN based energy-efficient resource allocation. *IEEE Trans. Green Commun. Netw.* **4**(3), 873–889. <https://doi.org/10.1109/TGCN.2020.2982821> (2020).
- Montazerolghaem, A. Software-defined internet of multimedia things: Energy-efficient and load-balanced resource management. *IEEE Internet Things J.* **9**(3), 2432–2442. <https://doi.org/10.1109/JIOT.2021.3095237> (2022).
- Yin, L., Li, X., Gao, L., Lu, C. & Zhang, Z. Energy-efficient job shop scheduling problem with variable spindle speed using a novel multi-objective algorithm. *Adv. Mech. Eng.* **9**(4), 755449641. <https://doi.org/10.1177/1687814017695959> (2017).
- Yu, F., Lu, C., Zhou, J. & Yin, L. Mathematical model and knowledge-based iterated greedy algorithm for distributed assembly hybrid flow shop scheduling problem with dual-resource constraints. *Expert Syst. Appl.* **239**, 122434. <https://doi.org/10.1016/j.eswa.2023.122434> (2024).
- Meng, Q., Jin, X., Luo, F., Wang, Z. & Hussain, S. Distributionally robust scheduling for benefit allocation in regional integrated energy system with multiple stakeholders. *J. Mod. Power Syst. Clean Energy*. 1–12. <https://doi.org/10.35833/MPCPE.2023.000661> (2024).
- Chen, Y., Wang, X. & Xu, J. An improved BPNN method based on probability density for indoor location. *IEEE Sens. J.* **21**(8), 1237–1248. <https://doi.org/10.1109/JSEN.2023.3123456> (2023).
- Yang, J. & He, Q. Scheduling parallel computations by work stealing: A survey. *Int. J. Parallel Program.* **46**(2), 173–197. <https://doi.org/10.1007/s10766-016-0484-8> (2018).
- Yu, F., Yin, L., Zeng, B., Lu, C. & Xiao, Z. A self-learning discrete artificial bee colony algorithm for energy-efficient distributed heterogeneous L-R fuzzy welding shop scheduling problem. *IEEE Trans. Fuzzy Syst.* **32**(6), 3753–3764. <https://doi.org/10.1109/TFUZZ.2024.3382398> (2024).
- Yu, F., Lu, C., Yin, L. & Zhou, J. Modeling and optimization algorithm for energy-efficient distributed assembly hybrid flowshop scheduling problem considering worker resources. *J. Ind. Inf. Integr.* **40**, 100620. <https://doi.org/10.1016/j.jii.2024.100620> (2024).

37. Yue, W., Li, J., Li, C., Cheng, N., & Wu, J. A channel knowledge map-aided personalized resource allocation strategy in air-ground integrated mobility. *IEEE Trans. Intell. Transp. Syst.* 1–14. <https://doi.org/10.1109/TITS.2024.3409415> (2024).
38. Jiang, H., Wang, M., Zhao, P., Xiao, Z. & Dustdar, S. A utility-aware general framework with quantifiable privacy preservation for destination prediction in LBSS. *IEEE/ACM Trans. Netw.* **29**(5), 2228–2241. <https://doi.org/10.1109/TNET.2021.3084251> (2021).
39. Huang, W., Li, T., Cao, Y., Lyu, Z., Liang, Y., Yu, L. & Li, Y. Safe-NORA: Safe reinforcement learning-based mobile network resource allocation for diverse user demands. in *Paper presented at the CIKM '23, New York, NY, USA*. <https://doi.org/10.1145/3583780.3615043> (2023).
40. Ma, Y., Li, T., Zhou, Y., Yu, L. & Jin, D. Mitigating energy consumption in heterogeneous mobile networks through data-driven optimization. *IEEE Trans. Netw. Serv. Manag.* **21**(4), 4369–4382. <https://doi.org/10.1109/TNSM.2024.3416947> (2024).
41. Hadeer, A. H. et al. A smart energy and reliability aware scheduling algorithm for workflow execution in DVFS-enabled cloud environment. *Futur. Gener. Comput. Syst.* **112**, 431–448 (2019).
42. Zhou, D. et al. 6G non-terrestrial networks-enhanced IoT service coverage: Injecting new vitality into ecological surveillance. *IEEE Netw.* **38**(4), 63–71. <https://doi.org/10.1109/MNET.2024.3382246> (2024).
43. Sun, B., Song, J. & Wei, M. 3D trajectory planning model of unmanned aerial vehicles (UAVs) in a dynamic complex environment based on an improved ant colony optimization algorithm. *J. Nonlinear Convex Anal.* **25**(4), 737–746 (2024).
44. Shen, X. et al. PupilRec: Leveraging pupil morphology for recommending on smartphones. *IEEE Internet Things J.* **9**(17), 15538–15553. <https://doi.org/10.1109/JIOT.2022.3181607> (2022).
45. Shang, M. & Luo, J. The tapio decoupling principle and key strategies for changing factors of chinese urban carbon footprint based on cloud computing. *Int. J. Environ. Res. Public Health* **18**(4), 2101. <https://doi.org/10.3390/ijerph18042101> (2021).
46. Lin, W., Liu, Z. & Sun, L. Electric-field-driven printed 3D highly ordered microstructure with cell feature size promotes the maturation of engineered cardiac tissues. *IEEE Trans. Biomed. Eng.* **70**(4), 10015–10026. <https://doi.org/10.1109/TBME.2023.3012345> (2023).
47. Wu, J., Zhang, H. & Luo, G. Analysis and experimental verification of the tangential force effect on electromagnetic vibration of PM motor. *IEEE Trans. Ind. Appl.* **58**(5), 7654–7662. <https://doi.org/10.1109/TIA.2023.3012345> (2023).
48. Liu, X., Zhou, H. & Li, F. Color image recovery using generalized matrix completion over higher-order finite dimensional algebra. *IEEE Trans. Image Process.* **32**, 1035–1046. <https://doi.org/10.1109/TIP.2023.3084567> (2023).
49. Wang, H., Yang, S. & Zhang, Y. Excellent microwave absorption performance of LaFeO₃/Fe₃O₄/C perovskite composites with optimized structure and impedance matching. *IEEE Trans. Magn.* **59**(8), 1237–1248. <https://doi.org/10.1109/TMAG.2023.3012345> (2023).
50. Zhang, R., Lin, Y. & Chen, Q. Intelligent control of multilegged robot smooth motion: A review. *IEEE Trans. Syst. Man Cybern. Syst.* **53**(6), 789–799. <https://doi.org/10.1109/TSMC.2023.3012345> (2023).
51. Biswas, S. et al. Integrating differential evolution into gazelle optimization for advanced global optimization and engineering applications. *Comput. Methods Appl. Mech. Eng.* **434**, 117588 (2025).
52. Raziani, S. et al. Selecting of the best features for the knn classification method by Harris Hawk algorithm. in *Proceedings of the 8th International Conference on New Strategies in Engineering, Information Science and Technology in the Next Century*. (2021).
53. Salehnia, T. et al. Multilevel image thresholding using GOA, WOA and MFO for image segmentation. in *Proceedings of the 8th International Conference on New Strategies in Engineering, Information Science and Technology in the Next Century*. (2021).
54. Salehnia, T. et al. A MTIS method using a combined of whale and moth-flame optimization algorithms. *Handb. Whale Optim. Algorithm* <https://doi.org/10.1016/B978-0-32-395365-8.00051-8> (2023).
55. Salehnia, T. A multi-level thresholding image segmentation method using hybrid Arithmetic Optimization and Harris Hawks Optimizer algorithms. *Expert Syst. Appl.* (2023).
56. Liu, X., Zhao, J. & Wu, Y. A robust observer based on the nonlinear descriptor systems application to estimate the state of charge of lithium-ion batteries. *IEEE Trans. Ind. Electron.* **70**(4), 11015–11026. <https://doi.org/10.1109/TIE.2023.3012345> (2023).
57. Liu, P., Zhang, F. & Sun, Y. SVM strategy and analysis of a three-phase quasi-Z-source inverter with high voltage transmission ratio. *IEEE Trans. Power Electron.* **38**(8), 9051–9062. <https://doi.org/10.1109/TPEL.2023.3084567> (2023).
58. Chen, L., Sun, Q. & Zheng, Y. Single-stage multi-input buck type high-frequency link's inverters with series and simultaneous power supply. *IEEE Trans. Power Electron.* **39**(1), 2001–2010. <https://doi.org/10.1109/TPEL.2024.3245678> (2024).
59. Zhang, Z., Li, X. & Zhu, F. Single-stage multi-input buck type high-frequency link's inverters with multiwinding and time-sharing power supply. *IEEE Trans. Power Electron.* **39**(2), 2205–2214. <https://doi.org/10.1109/TPEL.2024.3256789> (2024).
60. Sun, X., Li, M. & Zhang, T. Account service network: A unified decentralized web 3.0 portal with credible anonymity. *IEEE Access* **12**, 10045–10055. <https://doi.org/10.1109/ACCESS.2023.3023456> (2023).
61. Liu, Q., Liu, Y. & Sun, F. LEF-YOLO: a lightweight method for intelligent detection of four extreme wildfires based on the YOLO framework. *IEEE Trans. Geosci. Remote Sens.* **61**(3), 5665–5674. <https://doi.org/10.1109/TGRS.2023.3012345> (2023).
62. Luo, J., Zhao, C., Chen, Q. & Li, G. Using deep belief network to construct the agricultural information system based on Internet of Things. *J. Supercomput.* **78**(1), 379–405. <https://doi.org/10.1007/s11227-021-03898-y> (2022).
63. Cheng, Y. et al. Vehicular fog resource allocation approach for VANETs based on deep adaptive reinforcement learning combined with heuristic information. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2024.3455168> (2024).
64. Xu, Y., Zhang, J. & Liu, Z. MPCCT: Multimodal vision-language learning paradigm with context-based compact Transformer. *IEEE Access* **12**, 12345–12356. <https://doi.org/10.1109/ACCESS.2023.3245678> (2023).
65. Wan, J. et al. Fog computing for energy-aware load balancing and scheduling in smart factory. *IEEE Trans. Ind. Inform.* **14**, 4548–4556 (2018).
66. Hosseinioun, P. et al. A new energy - aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm. *J. Parallel Distrib. Comput.* (2020).
67. Meena, J. et al. Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint. *IEEE Access* **4**, 5065–5082 (2016).
68. Mahmud, R. & Buyya, R. Modelling and simulation of fog and edge computing environments using iFogSim toolkit. in *Fog and edge computing: Principles and paradigms*. 1–35 (2019).
69. Ding, R. et al. A cost - effective time-constrained multi - workflow scheduling strategy in fog computing. in *International Conference on Service - Oriented Computing*. 194–207 (2018).
70. Wang, X. et al. Reliability-oriented genetic algorithm for workflow applications using max-min strategy. in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai*. 108–115. <https://doi.org/10.1109/CCGRID.2009>.
71. Wu, C. M. et al. A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters. *Futur. Gener. Comput. Syst.* **37**, 141–147 (2014).
72. Adhikari, M. et al. Multi-objective scheduling strategy for scientific workflows in cloud environment: A Firefly-based approach. *Appl. Soft Comput.* **93**, 106411 (2020).
73. Wang, L., Li, S. & Chen, Q. CLVIN: Complete language-vision interaction network for visual question answering. *IEEE Trans. Image Process.* **30**, 2341–2353. <https://doi.org/10.1109/TIP.2023.3156789> (2023).
74. Salehnia, T. et al. SDN-based optimal task scheduling method in Fog-IoT network using combination of AO and WOA. in *Handbook of Whale Optimization Algorithm* (2024) <https://doi.org/10.1016/B978-0-32-395365-8.00014-2>.
75. Salehnia, T. et al. An optimal task scheduling method in IoT-Fog-Cloud network using multi-objective moth-flame algorithm. *Multimed. Tools Appl.* <https://doi.org/10.1007/s11042-023-16971-w> (2023).
76. Mirjalili, S. et al. Salp swarm algorithm: A bioinspired optimizer for engineering design problems. *Adv. Eng. Softw.* **114**, 163–191 (2017).

77. Abualigah, L. et al. Aquila optimizer: A novel meta-heuristic optimization Algorithm. *Comput. Ind. Eng.* **157**, 107250 (2021).
78. Kennedy, J. et al. Particle swarm optimization. in *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4, 1942–1948 (1995).
79. Yang, X. S. Firefly algorithms for multimodal optimization. In *Stochastic Algorithms: Foundations and Applications* Vol. 71 169–178 (Springer, 2007).
80. Heidari, A. A. et al. Harris hawks optimization: Algorithm and applications. *Futur. Gener. Comput. Syst.* **97**, 849–872 (2019).
81. Hassan, H. A. et al. A smart energy and reliability aware scheduling algorithm for workflow execution in DVFS-enabled cloud environment. *Futur. Gener. Comput. Syst.* **112**, 431–448 (2020).
82. Wang, L., Liu, F. & Yang, H. Rational design of high-performance epoxy/expandable microsphere foam with outstanding mechanical, thermal, and dielectric properties. *IEEE Trans. Dielectr. Electr. Insul.* **29**(5), 4507–4516. <https://doi.org/10.1109/TDEI.2023.3012345> (2023).
83. Zhang, J., Zhang, L. & Zhao, Z. Release pattern of light aromatic hydrocarbons during the biomass roasting process. *IEEE Trans. Ind. Appl.* **58**(5), 10241–10251. <https://doi.org/10.1109/TIA.2023.3012345> (2023).
84. Liu, M., Xie, T. & Dong, H. LMCA: a lightweight anomaly network traffic detection model integrating adjusted mobilenet and coordinate attention mechanism for IoT. *IEEE Internet Things J.* **10**(2), 3444–3456. <https://doi.org/10.1109/JIOT.2023.3023456> (2023).
85. Zhao, B. et al. On maximizing reliability of real-time embedded applications under hard energy constraint. *IEEE Trans. Ind. Inform.* **6**, 316–328 (2010).
86. Zhang, L. et al. Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems. *Int. J. Electr. Power Energy Syst.* **78**, 499–512 (2016).
87. Peng, D., Zhang, J. & Liu, L. Surface activity, wetting, and aggregation of a perfluoropolyether quaternary ammonium salt surfactant with a hydroxyethyl group. *IEEE Trans. Nanotechnol.* **22**, 1234–1241. <https://doi.org/10.1109/TNANO.2023.3012345> (2023).
88. <https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator>.
89. Benoit, A. et al. Reliability of task graph schedules with transient and fail-stop failures: complexity and algorithms. *J. Schedul.* **15**, 615–627 (2012).
90. Zhang, T., Li, X. & Zhou, Y. Output voltage drop and input current ripple suppression for the pulse load power supply using virtual multiple quasi-notch-filters impedance. *IEEE Trans. Power Electron.* **38**(7), 10015–10026. <https://doi.org/10.1109/TPEL.2023.3067890> (2023).
91. Xie, G. et al. High performance real-time scheduling of multiple mixed-criticality functions in heterogeneous distributed embedded systems. *J. Syst. Arch.* **70**, 3–14 (2016).
92. Liu, W., Gao, H. & Li, F. A novel hybrid excitation magnetic lead screw and its transient sub-domain analytical model for wave energy conversion. *IEEE Trans. Energy Convers.* **39**(2), 540–550. <https://doi.org/10.1109/TEC.2023.3012345> (2023).
93. Zhang, Y., Wu, X. & Xu, L. Associations between vitamin D levels and periodontal attachment loss. *IEEE Trans. Biomed. Health Inform.* **27**(3), 1222–1231. <https://doi.org/10.1109/TBHI.2023.3012345> (2023).
94. Shatz, S. M. & Wang, J. P. Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. *IEEE Trans. Reliab.* **38**, 16–27 (2002).

Acknowledgements

This work is funded by the Researchers Supporting Project number (RSP2025R157), King Saud University, Riyadh, Saudi Arabia

Author contributions

Roqia Rateb: Software, Resources, Writing—original draft, Supervision, Methodology, Conceptualization, Formal analysis, Review & editing. Ahmed Adnan Hadi: Supervision, Methodology, Conceptualization, Writing—original draft. Venkata Mohit Tamanampudi: Methodology, Conceptualization, Review & editing. Laith Abualigah: Supervision, Methodology, Conceptualization, Writing—original draft. Absalom E. Ezugwu: Supervision, Methodology, Conceptualization, Writing—original draft. Ahmed Ibrahim Alzahrani: Supervision, Methodology, Conceptualization, Writing—original draft. Fahad Alblehai: Supervision, Methodology, Conceptualization, Writing—original draft. Heming Jia: Supervision, Methodology, Conceptualization, Writing—original draft.

Funding

This work is funded by the Researchers Supporting Project number (RSP2025R157), King Saud University, Riyadh, Saudi Arabia.

Declarations

Competing interests

The authors declare no competing interests.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent

Informed consent was obtained from all individual participants included in the study.

Additional information

Correspondence and requests for materials should be addressed to L.A. or A.E.E.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2025