Expert Systems With Applications 228 (2023) 120401

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Efficient, economical and energy-saving multi-workflow scheduling in hybrid cloud[☆]

Zaixing Sun [a,b], Hejiao Huang [a,b], Zhikai Li [a,b], Chonglin Gu [a,b,*], Ruitao Xie [c], Bin Qian [d]

[a] School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518000, China
[b] Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen 518000, China
[c] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518000, China
[d] School of Information Engineering and Automation, Kunming University of Science and Technology, Kunming 650500, China

ARTICLE INFO

Keywords:
Hybrid cloud
Multi-workflow scheduling
Directed acyclic graph
Heuristic algorithm
Multi-objective optimization
Pareto front

ABSTRACT

Benefiting from the flexible, scalable and secure environment, hybrid cloud can overcome the shortage of limited resources in private cloud to simultaneously execute large-scale scientific workflows. In hybrid cloud, privacy-sensitive tasks are not allowed to be executed on public resources, while non-sensitive tasks are unrestricted. As an NP-Complete problem, it is extraordinarily challenging to schedule multiple workflows efficiently, economically and energy-savingly under quality-of-service constraints. This paper models the hybrid-cloud-based privacy-aware multi-workflow scheduling as a tri-objective optimization problem that optimizes workflow-oriented total tardiness, private-cloud-oriented total energy consumption, and public-cloud-oriented total monetary cost. To the best of authors' knowledge, few studies have been conducted on the tri-objective privacy-aware multi-workflow scheduling in hybrid cloud (PMWS-HC). To solve this problem, we dissect various factors involved during task scheduling and devise a novel Heuristic Scheduling Algorithm based on 9 Factors (HSA9Fs), which dynamically selects the workflows and tasks to be scheduled, and the corresponding VMs to execute them. To optimize the three conflicting objectives simultaneously, we propose a nested algorithm called MSIA, which first employs a Multi-objective Salp swarm algorithm to explore for the Pareto solutions, and then uses an Iterative greedy Algorithm to perform a refined search on individuals to obtain high-quality solutions. Extensive Medium-Small-Scale and Large-Scale simulation experiments show that both HSA9Fs and MSIA outperform state-of-the-art scheduling algorithms in several multi-objective performance metrics.

1. Introduction

Hybrid cloud is a primary computing paradigm that enables isolation, sharing, processing and storing of resources across private and public clouds to control costs and achieve an optimal balance among privacy, scalability and elasticity. Based on this, enterprises can choose whenever and wherever to host and process data to satisfy performance and cost requirements. Furthermore, the enterprises with hybrid cloud can quickly adapt to business and technological changes. Therefore, how to efficiently manage hybrid cloud resources and schedule tasks with users' requirements is an urgent issue to address.

Workflow originates from the manufacturing industry and has been widely applied in IT, management, transportation and other industries (Deelman et al., 2009; van der Aalst & van Hee, 2004). Nowadays, scientific workflow is growing increasingly complicated, with hundreds or even thousands of various types of tasks with dependencies (Chen et al., 2019; Qin et al., 2022). Workflow applications typically have deadline constraints and need a large amount of computing resources to process. Furthermore, certain tasks, such as sensitive tasks of user information, financial or medical information related to privacy, must be executed on devices with high security. In reality, such sensitive tasks have been widespread (Ali et al., 2015; Mthunzi et al., 2020). The privacy-aware workflow is more suitable for execution in hybrid

☆ This work is financially supported by Shenzhen Science and Technology Program, China under Grant No. GXWD20220817124827001 and No. JCYJ20210324132406016, National Natural Science Foundation of China under Grant No. 61732022, and Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, China under Grant No. 2022B1212010005.

* Corresponding author at: School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518000, China.
E-mail addresses: szx_1010@stu.hit.edu.cn (Z. Sun), huanghejiao@hit.edu.cn (H. Huang), 20S051007@stu.hit.edu.cn (Z. Li), guchonglin@hit.edu.cn (C. Gu), drtxie@gmail.com (R. Xie), bin.qian@vip.163.com (B. Qian).

https://doi.org/10.1016/j.eswa.2023.120401
Received 15 February 2023; Received in revised form 5 May 2023; Accepted 5 May 2023
Available online 11 May 2023

cloud and has been preliminary studied (Lei et al., 2022; Sharif et al., 2017; Wen et al., 2020). For hybrid cloud, users must perform sensitive tasks in private cloud. When private resources are insufficient to meet quality of service (QoS) for non-sensitive tasks, near-unlimited public cloud resources can be rented on demand (Yuan et al., 2017). Cloud computing, as we all know, is a multi-user platform that can manage workflows submitted by multiple users simultaneously, leading to the emergence of Workflow as a Service (Hilman et al., 2020; Zhou et al., 2016). Compared with single workflow scheduling, the advantage of multiple workflow scheduling lies in sharing and reusing idle time slots. However, under the heterogeneous computing environment and different QoS constraints, multi-workflow scheduling becomes more complicated and difficult.

This paper focuses on privacy-aware multi-workflow scheduling in hybrid cloud (PMWS-HC), which is to schedule each task in workflows to an appropriate resource for satisfying some performance criteria. First, tardiness is a delay penalty if the workflow is completed after its deadline (Rimal & Maier, 2017). It is usually used to measure the time efficiency of executing a workflow. Since workflow scheduling problem has NP-complete property (Mohammad Hasani Zade et al., 2021; Wu & Wang, 2018; Xia et al., 2022) and it can be reduced to PMWS-HC, PMWS-HC is also NP-complete. Second, minimizing the cost of leasing public cloud resources under deadline constraints is an essential economic metric. A task being assigned to a high-capacity resource type usually involves a lower execution time and higher cost. Furthermore, when more VMs are used, tasks are scattered over VMs, increasing the idle time of VMs and data transmission time among tasks. Third, minimizing energy consumption has always been a significant issue that cannot be ignored in data centers. Tasks being assigned to low configuration resource types usually take longer to execute and consume less energy. Obviously, minimizing tardiness, leasing cost and energy consumption are three conflicting objectives, so PMWS-HC is a multi-objective optimization problem (MOP).

In recent decades, MOP is usually solved by evolutionary algorithms. The concept of Pareto domination is generally used in MOP to reflect the simultaneous trade-off among multiple objectives. Among the studies related to PMWS-HC, there are some studies (Pasdar et al., 2020; Rimal & Maier, 2017; Wang et al., 2021) that evaluate the algorithm performance on each objective separately. In this way, their algorithm is still a single-objective optimization algorithm, and cannot trade off multiple objectives simultaneously. Other studies (Hafsi et al., 2022; Li et al., 2022, 2019; Wang & Zuo, 2021; Xia et al., 2022; Zhou et al., 2019) directly specify the resources to execute tasks, ignoring the characteristics of workflow scheduling. This method relies entirely on algorithmic optimization and cannot proactively select the workflow and its tasks to be scheduled, as well as the resources needed to execute the task. Though there exist some studies (Chen et al., 2019; Saeedi et al., 2020; Wen et al., 2020) that involve simple rules to make decisions, they are limited to the initialization of algorithm, while the aforementioned decisions are still not proactively made during the optimization of the algorithm. There are many factors that should not be ignored during workflow scheduling. These factors can help us make decisions, such as (1) unscheduled tasks: their tightness relative to the deadline affects which task or workflow is scheduled first; (2) the task's actual available time: it will vary on different VMs and determine the task's actual start time, and it affects whether the scheduling scheme is compact or meets the QoS; (3) dependency between tasks: deploying a task to the VM that executes its parent task can save transmission time. Such factors need to be further explored and fully utilized to make scheduling decisions, but none of the existing studies have done so.

In order to make scheduling decisions proactively, we dissect the scheduling process to extract 9 influencing factors (i.e., unscheduled levels, current completion times, workflow deadline, existing scheduling results, urgency, sub-deadline, Earliest Available Time, Actual Available Time, and the maximum completion time of all scheduled tasks). Based on this, we devise a novel Heuristic Scheduling Algorithm based on 9 Factors (HSA9Fs), which can overcome the above shortcomings of the existing algorithms. The heuristic algorithm has a limited search space due to its strong regularity, whereas MOP has many non-dominated solutions. Therefore, employing the global search capabilities of evolutionary algorithm, we propose a nested algorithm of Multi-objective Salp swarm algorithm (MSSA) (Mirjalili et al., 2017) and Iterative greedy Algorithm (IGA) (Ruiz & Stützle, 2007), named MSIA, to further tackle this problem. Since MSIA's decoding is based on HSA9Fs which can quickly evaluate individuals, MSIA can solve large-scale problems. To the best of our knowledge, few studies have been conducted on the tri-objective PMWS-HC. Our main contributions are listed as follows:

- We establish a hybrid-cloud-based privacy-aware multi-workflow scheduling model, which simultaneously considers minimizing workflow-oriented total tardiness, private-cloud-oriented total energy consumption and public-cloud-oriented total monetary cost.
- By dissecting various factors involved during scheduling, we creatively propose HSA9Fs, which dynamically selects the workflows and tasks to be scheduled and the VMs to be executed. Our scheme is non-greedy and can schedule tasks compactly by comprehensively considering 9 factors.
- To explore Pareto solutions for trading off the tri-objective considered, we exploit MSSA to search Pareto solution globally to achieve the joint optimization of efficiency, economy and energy-savingly and IGA to deeply search in the neighborhoods of individual to improve the quality of the solution.
- We conduct extensive Medium-Small-Scale and Large-Scale simulation experiments based on five well-known real-world workflow applications to investigate the diversity, convergence and efficiency of the proposed algorithms. The results show that both HSA9Fs and MSIA outperform state-of-the-art scheduling algorithms in several multi-objective performance metrics.

The organization of this paper is as follows: In Section 2, we review the related work on privacy-aware workflow and multi-objective workflow scheduling. In Section 3, we present the cloud workflow scheduling model. In Section 4, the proposed HSA9Fs and MSIA are presented in detail. Section 5 verifies the performance of the proposed algorithms. Section 6 concludes the paper.

2. Related work

Security and privacy protection have become key issues in cloud environments (Mthunzi et al., 2020; Ren et al., 2019). As a typical cloud application, workflow scheduling also has related issues and some research as reported in Hu et al. (2020), Lei et al. (2022), Li et al. (2016). Li et al. (2016) considered security overhead model and made risk analysis for workflow, and proposed a security and cost aware scheduling algorithm for workflow in public cloud, which tried to minimize the total workflow execution cost while meeting the deadline and risk rate constraints. Hu et al. (2020) considered a privacy-aware Spark application scheduling problem in hybrid cloud and proposed a scheduling algorithm SSPPH (Spark Scheduling with Privacy Protection in a Hybrid Cloud) to minimize the total rental cost. Spark application is modeled as a two-layer topological structure. Lei et al. (2022) proposed two scheduling heuristics for privacy and security-aware workflow scheduling in hybrid cloud to optimize the cost under the constraints of deadline and privacy. They developed a hybrid encryption method based on three levels constraints to ensure data transmission security.

These previous studies focused on single-objective optimization, but there are also related investigations involving multi-objective optimization. Wang et al. (2021) considered security-aware bag-of-tasks scheduling problem in hybrid cloud. They proposed a heuristic task scheduling method concerning Security and evaluated the method on

Table 1
Comparison of related works for workflow scheduling problem.

Ref.	Workflow model	Resource model	Optimization model	Problem's characteristics
Li et al. (2016)	Single workflow	Public cloud	1: Cost	Security overhead model and workflow risk analysis.
Hu et al. (2020)	Spark application	Hybrid cloud	1: Cost	Privacy tasks can only be executed on private cloud resources.
Lei et al. (2022)	Single workflow	Hybrid cloud	1: Cost	Encryption and decryption processes are integrated into the scheduling model, which is similar to the transmission time.
Rimal and Maier (2017)	Multi-workflow	Public cloud	[a]: Makespan, Tardiness and Resource utilization rate	The cost of data transmission is not considered.
Pasdar et al. (2020)	Single workflow	Hybrid clouds	[a]: Cost and Makespan	Overall execution cost depends on the cost of used storage, consumed bandwidth, and computation in public cloud.
Wang et al. (2021)	Bag-of-tasks	Hybrid clouds	[a]: Cost, The number of finished tasks and Energy	Each task has corresponding deadline and security level constraints, and a security model is established.
Wen et al. (2020)	Single workflow	Public clouds	2: Cost and Makespan	The multi-data center model and storage cost are considered.
Hafsi et al. (2022), Zhou et al. (2019)	Single workflow	Hybrid clouds	2: Cost and Makespan	Trade-off between makespan and cost.
Xia et al. (2022)	Single workflow	Private cloud	2: Makespan and Energy	Trade-off between makespan and energy consumption.
Li et al. (2019)	Multi-workflow	Public cloud	3: Cost, Makespan and Service quality	Task scheduling in cloud manufacturing; Maximize service quality.
Li et al. (2022)	Multi-workflow	Hybrid clouds	3: Cost, Makespan and Energy	Both on-demand and reserved instance types are considered; Energy saving by Dynamic Voltage Frequency Scaling.
Saeedi et al. (2020)	Single workflow	Public cloud	4: Cost, Makespan, Energy and Reliability	Energy consumption only considers dynamic energy consumption.

[a]This multi-objective optimization problem does not evaluate the performance of the algorithm through the non-dominated solution, but each objective is evaluated and analyzed separately.

metrics such as makespan, cost efficiency and energy efficiency. Pasdar et al. (2020) proposed a two-stage algorithm to minimize execution time and cost: it first generated an initial scheduling strategy based on an extended genetic algorithm (GA) and then dynamically adjusted reschedule some tasks in response to volatile execution environments. Rimal and Maier (2017) proposed a cloud-based workflow scheduling policy for batch compute-intensive workflows in multi-tenant cloud computing. They evaluated the proposed algorithm in terms of makespan, tardiness and resource utilization rate, and etc. However, the above works are limited to evaluating the considered objectives separately, rather than trading off multiple objectives. Nondominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002) is a classical Pareto-Dominance-based multi-objective optimization algorithm. Wen et al. (2020) proposed a multi-objective scheduling algorithm based on GA for workflow scheduling with privacy protection constraints. The trade-off objectives are makespan and monetary cost. Based on many-objective particle swarm optimization (MaOPSO) (Figueiredo et al., 2016), Saeedi et al. (2020) proposed an improved MaOPSO to optimize multiple conflicting objectives including maximization of reliability and minimization of cost, makespan and energy consumption. Zhou et al. (2019) and Hafsi et al. (2022) proposed multi-objective approach named MOH (Multi-objective Optimization for Hybrid clouds) and Genetically-modified Multi-objective Particle Swarm Optimization (GMPSO), respectively, to simultaneously optimize makespan and cost in hybrid clouds. Xia et al. (2022) proposed a multi-objective GA combined with longest common subsequence, named GALCS, to simultaneously optimize makespan and energy consumption in heterogeneous cloud. Integrated longest common subsequence can record the beneficial gene blocks and increase GA performance.

There are also a few works focusing on multi-objective multi-workflow scheduling. Li et al. (2019) proposed two multi-objective algorithms, ant colony optimization-based multi-objective algorithm (MACO) and NSGA-II-based multi-objective algorithm (MGA), to solve the multi-task scheduling in cloud manufacturing. They developed a model for scheduling many heterogeneous complicated tasks while taking into account three objectives: makespan, total cost, and total service quality. In their model, each task is represented by a DAG (Directed Acyclic Graph), and all tasks are available at the beginning with the same priorities. Therefore, it can be regarded as a tri-objective multi-workflow scheduling problem. Li et al. (2022) proposed a Chaotic-nondominated-sorting Owl Search Algorithm to solve multi-workflow scheduling in hybrid clouds. The optimization objective was to minimize makespan, cost and energy consumption simultaneously under the deadline and budget constraints. They considered both reserved and on-demand pricing models in the public cloud and utilized Dynamic Voltage Frequency Scaling to reduce energy consumption and save costs. Pan et al. (2021) proposed a Multi-objective Clustering Evolutionary Algorithm (based on Strength Pareto Evolutionary Algorithm He et al., 2017) for multi-workflow scheduling under deadline constraint in Mobile Edge Computing to minimize the cost and energy consumption.

The details of the above related works for workflow scheduling problem are tabulated in Table 1.

3. Cloud workflow scheduling model

Fig. 1 presents a multi-workflow scheduling framework in hybrid cloud. Users submit workflows through the Workflow as a Service Portal within a certain time interval. The Cloud Resource Manager monitors the resources in hybrid cloud, and then cooperates with the Task Scheduler to complete the tasks-to-resources mapping, and feeds back the scheduling results to users. Task scheduler is the core of the framework. This paper focuses on the tri-objective optimization of offline multi-workflow in hybrid cloud. The preemption of tasks is not allowed.

This section first introduces the structure and data related to workflow application, and then introduces the different configurations of public cloud and private cloud, and finally constructs a tri-objective multi-workflow scheduling model through workflow scheduling analysis. Table 2 summarizes the symbols used in this section to improve readability.

3.1. Privacy-aware multi-workflow applications model

In cloud, a set of workflow applications can be represented as $G = \left\{ G^g \middle| g = 1, 2, \ldots, |G| \right\}$, which $G^g = (A^g, W^g, E^g, D^g, \Phi^g)$ is the DAG description of a workflow.

- $A^g = \left\{ a_i^g \middle| i = 1, 2, \ldots, |A^g| \right\}$ is the set of tasks, where a_i^g denotes a task in workflow G^g and $|A^g|$ denotes the total number of tasks in the workflow.
- $W^g = \left\{ w_i^g \middle| a_i^g \in A^g \right\}$ is the set of weights on tasks, which denotes the computation of tasks in giga floating point operations (GFLOP).
- $E^g = \left\{ e_{ij}^g = \left(a_i^g, a_j^g \right) \middle| a_i^g, a_j^g \in A^g; i < j \right\}$ is the set of data dependency constraints between tasks. Let $Suc(a_i^g) = \left\{ a_j^g \middle| e_{ij}^g \in E^g \right\}$ and $Pre(a_i^g) = \left\{ a_j^g \middle| e_{ji}^g \in E^g \right\}$ be the sets of all immediate successors and predecessors of task a_i^g, respectively. A task a_i^g cannot be started

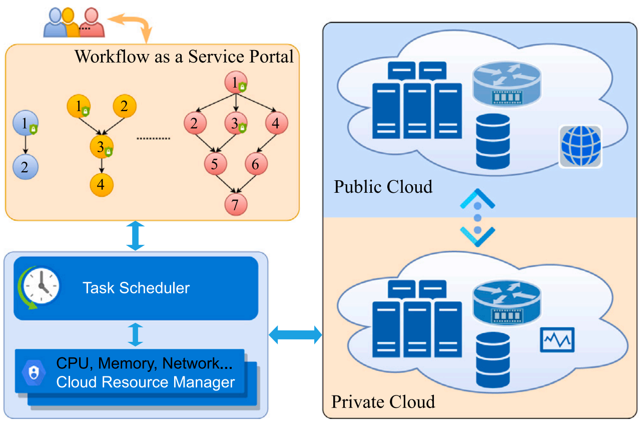


Fig. 1. Multi-workflow scheduling framework in hybrid cloud. 🛡 represents that the task is private sensitive.

unless all of its immediate predecessors (i.e. $Pre(a_i^g)$) have finished its execution and the relevant data dependencies have been transmitted to a_i^g.

- $D^g = \left\{ d_{ij}^g \middle| e_{ij}^g \in E^g \right\}$ is the set of transmitted data, where d_{ij}^g is the volume of data to be transmitted from task a_i^g to task a_j^g, in GFLOP.
- $\Phi^g = \left\{ \phi_i^g \middle| \phi_i^g \in \{0,1\}, a_i^g \in A^g \right\}$ is the set of privacy tags for tasks. If $\phi_i^g = 1$, it means that task a_i^g contains private data and can only be deployed on resource instances in private cloud; Otherwise, it can be deployed on resource instances in private or public clouds.

3.2. Hybrid cloud model

Hybrid cloud is made up of two cloud data centers: public cloud and private cloud. The computing resources in both data centers are in the form of virtual machines (VMs) or instances. Generally, private cloud provides a fixed number of VMs, whereas public cloud provides an "unlimited" number of VMs.

3.2.1. Public cloud

In public cloud, VM instances have the following characteristics: processing capacity, network bandwidth, leasing price. Let $P^0 = \left\{ p_k^0 \middle| k = 1, 2, \ldots, \left|P^0\right| \right\}$ denote the set of all instance types in public cloud, where $\left|P^0\right|$ is the total number of types.

- $U = \left\{ U(p_k^0) \middle| p_k^0 \in P^0 \right\}$ is the set of processing capacity of CPU in Giga Floating Point Operations Per Second (GFLOPS, a widely used metric Rodriguez & Buyya, 2014; Sahni & Vidyarthi, 2018), where $U(p_k^0)$ is the processing capacity of instance type p_k^0.
- $B = \left\{ B(p_k^0, p_h^0) \middle| p_k^0, p_h^0 \in P^0 \right\}$ is the set of communication bandwidth between different instance types. $B(p_k^0, p_h^0)$ is the communication bandwidth between instance type p_k^0 and p_h^0, which depends on the smaller bandwidth of the two instances (denoted as $b(p_k^0)$ and $b(p_h^0)$, respectively). That is, $B(p_k^0, p_h^0) = \min\left\{ b(p_k^0), b(p_h^0) \right\}$.
- $M = \left\{ M(p_k^0) \middle| p_k^0 \in P^0 \right\}$ is the set of leasing prices, where $M(p_k^0)$ is the per-unit price of instance type p_k^0.
- $\vec{M}$ is the price of transmitting unit data. In general, public clouds charge their network resources only for the data transferred out,[1].

[1] https://www.aliyun.com/product/ecs https://aws.amazon.com/cn/ec2/pricing/on-demand/

Table 2
Symbols and meanings.

Symbol[a]	Explanation
	Multiple Workflows
G^g	A workflow, $G^g \in G$.
a_i^g	A task in workflow G^g, $a_i^g \in A^g$.
w_i^g	The weight of task a_i^g, $w_i^g \in W^g$.
e_{ij}^g	Data dependency between task a_i^g and a_j^g, $e_{ij}^g \in E^g$.
$Suc(a_i^g)$	All immediate successors of task a_i^g.
$Pre(a_i^g)$	All immediate predecessors of task a_i^g.
d_{ij}^g	The volume of data to be transferred from task a_i^g to task a_j^g, $d_{ij}^g \in D^g$.
ϕ_i^g	The privacy tag of task a_i^g, $\phi_i^g \in \{0,1\}$, $\phi_i^g \in \Phi^g$.
	Hybrid Cloud
p_k^0	An instance type in public cloud, $p_k^0 \in P^0$.
p_k^1	An instance type in private cloud, $p_k^1 \in P^1$.
P	All instance type in hybrid cloud, $P = P^0 \cup P^1$.
$U(p_k^0)$	The processing capacity of instance type p_k^0.
$B(p_k^0, p_h^0)$	The bandwidth between instance type p_k^0 and p_h^0.
$M(p_k^0)$	The per-unit price of instance type p_k^0.
$\vec{M}$	The price of transmitting unit data.
M^H	The per-unit price in Hibernation state.
$\bar{O}(p_k^1)$	The idle power of instance type p_k^1.
$\tilde{O}(p_k^1)$	The dynamic power of instance type p_k^1.
$\vec{O}$	The transmission power of router.
	Scheduling Model
v_h	An instances leased by users, $v_h \in V$.
η_h	The mapping relationship between VM v_h and data centers, $\eta_h \in \{0,1\}$.
v_h^P	The instance v_h's type, $v_h^P \in P$.
ζ_{ih}^g	The mapping relationship between task a_i^g and VM v_h, $\zeta_{ih}^g \in \{0,1\}$.
ET_{ih}^g	The execution time of task a_i^g on VM v_h.
$CT_{ij}^{g,kh}$	The communication time when tasks a_i^g and a_j^g are assigned to VMs v_k and v_h, respectively.
FT_i^g	The actual finish time of task a_i^g.
ST_i^g	The actual start time of task a_i^g.
C^g	The complete time of workflow G^g.
$\mathbb{T}^g$	The tardiness of workflow G^g.
DL^g	The deadline of workflow G^g.
$\mathbb{T}$	The **total tardiness** of multi-workflow.
$\tilde{E}_{ih}^g$	The energy consumption of task a_i^g.
$\tilde{\mathbb{E}}$	The total dynamic energy consumption.
$\bar{\mathbb{E}}$	The total static energy consumption.
$\vec{E}_{ij}^{g,kh}$	The communication energy consumption between task a_i^g and a_j^g.
$\vec{\mathbb{E}}$	The total communication energy consumption.
$\mathbb{E}$	The **total energy consumption**.
$\overline{ST}_{h\hbar}$	The $\hbar$-th start time of the rented VM v_h in running state.
$\overline{FT}_{h\hbar}$	The $\hbar$-th end time of the rented VM v_h in running state.
$\overline{ST}_{h\hbar'}^H$	The $\hbar'$-th start time of the rented VM v_h in hibernate state.
$\overline{FT}_{h\hbar'}^H$	The $\hbar'$-th end time of the rented VM v_h in hibernate state.
$\tilde{\mathbb{M}}$	The total rental cost.
$\vec{\mathbb{M}}_{ij}^{g,kh}$	The communication cost between task a_i^g and a_j^g.
$\vec{\mathbb{M}}$	The total communication cost.
$\mathbb{M}$	The **total monetary cost**.
$\mathtt{AS}$	The **non-dominated solution set/archive set**.

[a] In this table, $g = 1, 2, \ldots, |G|$; $i, j = 1, 2, \ldots, |A^g|$; Instance type index $k, h = 1, 2, \ldots, \left|P^0\right|$ (or $\left|P^1\right|$); Instance index $k, h = 1, 2, \ldots, |V|$.

The pricing model is based on an on-demand billing scheme and the minimum billing period is 1 s with a mandatory-minimum of 60 s. As in Sun et al. (2022), we consider cold startup time, warm startup time and hibernation state. Cold startup time refers to the initial boot time after the instance is leased, during which instance initialization and task mirror deployment are performed. Warm startup time refers to the restart time after hibernation, which is less than the cold startup time. Hibernation state refers to the state of suspend-to-disk operating system for which a lower per-unit price, denoted as M^H, is charged.

3.2.2. Private cloud

Let $P^1 = \left\{ p_k^1 \middle| k = 1, 2, \dots, \left|P^1\right| \right\}$ denote the set of all instance types in private cloud, where $\left|P^1\right|$ is the total number of types. An instance has the following characteristics:

- The definition of processing capacity and bandwidth is the same as that of public cloud.
- $\bar{O} = \left\{ \bar{O}(p_k^1) \middle| p_k^1 \in P^1 \right\}$ is the set of idle power, where $\bar{O}(p_k^1)$ is the idle power of instance type p_k^1.
- $\tilde{O} = \left\{ \tilde{O}(p_k^1) \middle| p_k^1 \in P^1 \right\}$ is the set of dynamic power, where $\tilde{O}(p_k^1)$ is the dynamic power of instance type p_k^1.
- $\vec{O}$ is the transmission power of router. Suppose routers have the same power, regardless of the network topology.

There is a limit to the number of instances in private cloud. It is assumed that the instances of private cloud can be obtained and used at any time until all tasks are finished. However, it still takes time to deploy the task execution scenario or initialize instance, assuming that the time is the same as the cold startup time in public cloud. Assume that hybrid cloud has enough memory to execute workflows.

3.3. Problem formulation

Let $V = [v_1, v_2, \dots, v_{|V|}]$ be the VM instances used by all users, where $|V|$ is the total number of VMs. Let $\eta_h \in \{0, 1\}$ be the mapping relationship between VMs and data centers. $\eta_h = 1$ indicates that v_h is in private cloud, otherwise it is in public cloud.

$$\eta_h = \begin{cases} 1, & v_h \text{ is in private cloud,} \\ 0, & v_h \text{ is in public cloud.} \end{cases} \tag{1}$$

$v_h^P \in P$ $(P = P^0 \cup P^1)$ is instance v_h's type. Thus, $U(v_h^P)$ is instance v_h's processing capacity. $M(v_h^P)$ is instance v_h's per-unit price. The parameters of the corresponding power are the same.

Let $\zeta_{ih}^g \in \{0, 1\}$ be the mapping relationship between tasks and VMs, where $\zeta_{ih}^g = 1$ if task a_i^g is assigned to v_h, otherwise it is 0.

$$\zeta_{ih}^g = \begin{cases} 1, & \text{task } a_i^g \text{ is assigned to } v_h, \\ 0, & otherwise. \end{cases} \tag{2}$$

We need to ensure that each task is scheduled only once. At the same time, when a task has a private property, it should be deployed in a private cloud. The corresponding constraint is shown in Eq. (3).

$$\begin{cases} \sum_{v_h \in V} \zeta_{ih}^g \times \eta_h = 1, & \phi_i^g = 1, \\ \sum_{v_h \in V} \zeta_{ih}^g = 1, & \phi_i^g = 0. \end{cases} \tag{3}$$

3.3.1. Total tardiness

The total tardiness is the sum of tardiness of all workflows. The execution time of task a_i^g on VM v_h is

$$ET_{ih}^g = \frac{w_i^g}{U(v_h^P)} \times \zeta_{ih}^g, \tag{4}$$

When tasks a_i^g and a_j^g are assigned to VMs v_k and v_h, respectively, their communication time is

$$CT_{ij}^{g,kh} = \begin{cases} \dfrac{d_{ij}^g}{B(v_k^P, v_h^P)} \times \zeta_{ik}^g \times \zeta_{jh}^g, & v_k \neq v_h, \\ 0, & otherwise. \end{cases} \tag{5}$$

When v_k equals to v_h, the communication time on the same VM is 0. The precedence constraint between tasks is as follows:

$$FT_i^g + CT_{ij}^{g,kh} \leq ST_j^g, \; e_{ij} \in E^g, \tag{6}$$

$$FT_i^g = ST_i^g + \sum_{v_h \in V} ET_{ih}^g, \; a_i \in A^g, \tag{7}$$

where FT_i^g denotes the actual finish time of task a_i^g, and ST_j^g denotes the actual start time of task a_j^g. The complete time of workflow G^g is $C^g = \max_{a_i \in A^g} \left\{ FT_i^g \right\}$. The makespan of multi-workflow scheduling is $C_{max} = \max_{G^g \in G} \{C^g\}$. The tardiness of workflow G^g is

$$\mathbb{T}^g = \max\{0, C^g - DL^g\}, \tag{8}$$

where DL^g is the deadline of workflow G^g.

The **total tardiness** of multi-workflow is

$$\mathbb{T} = \sum_{G^g \in G} \mathbb{T}^g. \tag{9}$$

3.3.2. Total energy consumption

The total energy consumption includes the dynamic energy consumption during the task execution in private cloud, the static energy consumption during standby, and the communication energy consumption during data transmission within the private cloud and across data centers. The energy consumption for executing task a_i^g is

$$\tilde{E}_{ih}^g = ET_{ih}^g \times \tilde{O}(v_h^P) \times \zeta_{ih}^g \times \eta_h.$$

The total dynamic energy consumption is

$$\tilde{\mathbb{E}} = \sum_{G^g \in G} \sum_{v_h \in V} \sum_{a_i \in A^g} \tilde{E}_{ih}^g. \tag{10}$$

The total static energy consumption is

$$\bar{\mathbb{E}} = \sum_{v_h \in V} \left(C_{max} - \sum_{G^g \in G} \sum_{a_i \in A^g} ET_{ih}^g \times \eta_h \right) \times \bar{O}(v_h^P). \tag{11}$$

The communication energy consumption between task a_i^g and a_j^g is

$$\vec{E}_{ij}^{g,kh} = \begin{cases} CT_{ij}^{g,kh} \times \vec{O}, & \eta_k + \eta_h \geq 1, \\ 0, & otherwise. \end{cases}$$

The total communication energy consumption is

$$\vec{\mathbb{E}} = \sum_{G^g \in G} \sum_{v_k \in V} \sum_{v_h \in V} \sum_{e_{ij} \in E^g} \vec{E}_{ij}^{g,kh}. \tag{12}$$

The **total energy consumption** is

$$\mathbb{E} = \tilde{\mathbb{E}} + \bar{\mathbb{E}} + \vec{\mathbb{E}}. \tag{13}$$

3.3.3. Total monetary cost

The total monetary cost includes the cost of leasing VMs in public cloud and the communication cost when outgoing from public cloud. Let $\overline{ST}_{h\hbar}$ and $\overline{FT}_{h\hbar}$ be the $\hbar$-th start and end time of the rented VM v_h in running state, $\hbar \geq 1$. Let $\overline{ST}_{h\hbar'}^H$ and $\overline{FT}_{h\hbar'}^H$ be the $\hbar'$-th start and end time of the rented VM v_h in hibernate state, $\hbar' \geq 0$. The total rental cost of VMs is

$$\tilde{\mathbb{M}} = \sum_{v_h \in V,\, \eta_h = 0} \Big(M(v_h^P) \times \sum_{\hbar} (\overline{FT}_{h\hbar} - \overline{ST}_{h\hbar}) + M^H \times \sum_{\hbar'} (\overline{FT}_{h\hbar'}^H - \overline{ST}_{h\hbar'}^H) \Big). \tag{14}$$

The communication cost between task a_i^g and a_j^g is

$$\vec{\mathbb{M}}_{ij}^{g,kh} = \begin{cases} d_{ij}^g \times \vec{M} \times \zeta_{ik}^g \times \zeta_{jh}^g, & \eta_k = 0 \;\&\; \eta_h = 1, \\ 0, & otherwise. \end{cases}$$

The total communication cost is

$$\vec{\mathbb{M}} = \sum_{G^g \in G} \sum_{v_k \in V} \sum_{v_h \in V} \sum_{e_{ij} \in E^g} \vec{M}_{ij}^{g,kh}. \tag{15}$$

The **total monetary cost** is

$$\mathbb{M} = \tilde{\mathbb{M}} + \vec{\mathbb{M}}. \tag{16}$$

3.3.4. Scheduling model

The objectives of the studied problem is to minimize the total tardiness, total energy consumption and total monetary cost of privacy-aware multi-workflow scheduling. Based on the above discussions, the constrained optimization problem can be formulated as follows:

$$\textbf{Minimize} \quad \mathbb{T},\ \mathbb{E},\ \mathbb{M} \tag{17a}$$

$$\textbf{subject to}: \text{Eq. (3)}, \quad \forall a_i^g \in A^g,\ \forall G^g \in G, \tag{17b}$$

$$\text{Eq. (4)}, \quad \forall v_h \in V,\ \forall a_i^g \in A^g,\ \forall G^g \in G, \tag{17c}$$

$$\text{Eq. (5)}, \quad \forall v_k, v_h \in V,\ \forall e_{ij}^g \in E^g,\ \forall G^g \in G, \tag{17d}$$

$$\text{Eqs. (6), (7), (8)}, \quad \forall G^g \in G, \tag{17e}$$

$$\eta_h \in \{0,1\}, \quad \forall v_h \in V, \tag{17f}$$

$$\zeta_{ih}^g \in \{0,1\}, \quad \forall v_h \in V,\ \forall a_i^g \in A^g,\ \forall G^g \in G, \tag{17g}$$

where the decision variables are ζ_{ih}^g, η_h and ST_i^g ($\forall v_h \in V$, $\forall a_i^g \in A^g$, $\forall G^g \in G$).

3.4. Basic concepts of MOP

MOP concerns more than one objective simultaneously to explore a trade-off between the conflicting objectives (Wu & Wang, 2018). it is usually formulated as follows:

$$\begin{cases} \text{minimize } f_1(x), f_2(x), \dots, f_z(x), \\ \text{subject to: } x \in X, \end{cases}$$

where $f_1(x), f_2(x), \dots, f_z(x)$ are objectives to be minimized, and x is a vector of decision variables, and X is the set of feasible solution. In most cases, there may not be a solution that minimizes all objectives. Therefore, the Pareto optimal solutions are adopted as the results of MOP.

Definition 1 (***Pareto Dominance***)**.** A solution x_1 is considered to (Pareto) dominate solution x_2 (denoted as $x_1 \succ x_2$) if and only if (a) $\forall i \in \{1,2,\dots,z\}$: $f_i(x_1) \le f_i(x_2)$, and (b) $\exists i \in \{1,2,\dots,z\}$: $f_i(x_1) < f_i(x_2)$.

Definition 2 (***Pareto Front***)**.** A solution x is called a non-dominated solution or Pareto solution if it is not dominated by any other solution. The solution set containing all Pareto solutions is defined as the Pareto Front/Pareto set.

The multi-objective optimization algorithms are required to find a **non-dominated solution set/archive set** (AS). There are two main metrics to evaluate the obtained AS: (1) HyperVolume reflecting the diversity and convergence of AS. (2) Coverage Rate reflecting the dominance relationship between two archive sets.

4. The proposed multi-workflow scheduling algorithm (MSIA)

Fig. 2 shows the flow chart of our proposed algorithm. At first, the population is initialized with HSA9Fs and random method, and the archive set is initialized. Then, MSSA is used to update the population, and IGA is used to further optimize individuals. The relevant implementation details are introduced in the following sections. Table 3 summarizes the symbols used in this section.

4.1. Preprocessing operator

We present the preprocessing operators involved in the proposed algorithm for preprocessing multi-workflow structures and data.

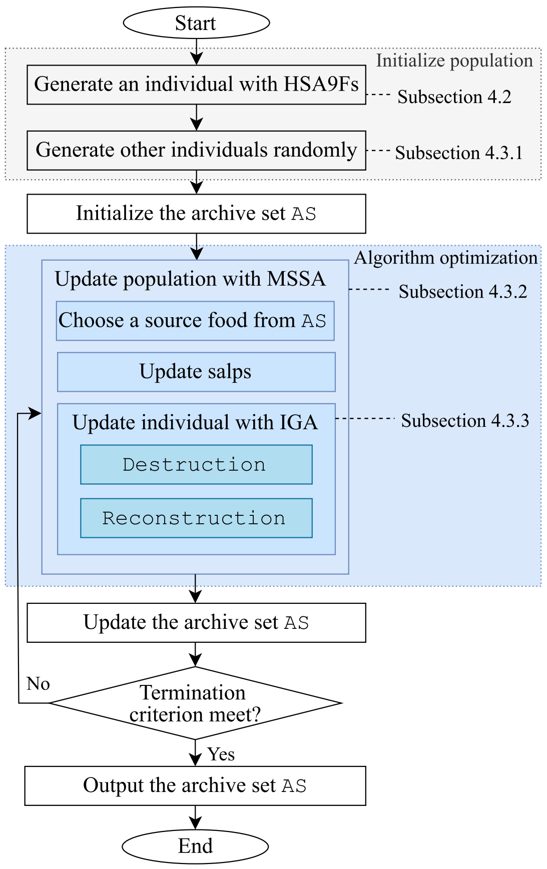

Start
Initialize population
Generate an individual with HSA9Fs
Subsection 4.2
Generate other individuals randomly
Subsection 4.3.1
Initialize the archive set AS
Algorithm optimization
Update population with MSSA
Subsection 4.3.2
Choose a source food from AS
Update salps
Update individual with IGA
Subsection 4.3.3
Destruction
Reconstruction
Update the archive set AS
No
Termination criterion meet?
Yes
Output the archive set AS
End

Fig. 2. Flow chart of the proposed MSIA.

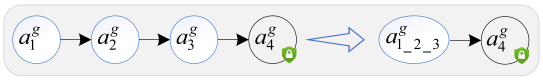


Fig. 3. An example of sequence tasks merging into one task block.

4.1.1. Merging

There are some sequential tasks with single-output and single-input structure in DAG, which satisfy Eq. (18). As in Sahni and Vidyarthi (2018) and Sun et al. (2022), we merge them into a task block to simplify DAG. The task block's execution time is the sum of the execution times of all of its internal tasks, and there is no data communication within the structure. Fig. 3 shows three sequence tasks merging into one task block.

$$\left|Suc(a_i^g)\right| = \sum_{a_j^g \in Suc(a_i^g)} \left|Pre(a_j^g)\right| = 1,\ \phi_i^g = \phi_j^g, \tag{18}$$

where $\phi_i^g = \phi_j^g$ denotes that the corresponding tasks have the same privacy tag.

4.1.2. Task topological level

As in Hu et al. (2020) and Sun et al. (2022), task a_i^g's topological level L_i^g in workflow G^g is

$$L_i^g = \begin{cases} 1, & if\ \ Pre(a_i^g) = \emptyset, \\ \max\limits_{a_j^g \in Pre(a_i^g)} \left\{ L_j^g \right\} + 1, & otherwise. \end{cases} \tag{19}$$

Thus, the set $\bar{a}_{li}^g \in \bar{A}^g$ of tasks at each level can be represented as

$$\bar{a}_l^g = \left\{ a_i^g \middle| l = L_i^g, a_i^g \in A^g \right\}, \tag{20}$$

Table 3
Symbols and meanings in Section 4.

Symbol	Definition
L_i^g	Task a_i^g's topological level.
$\bar{a}_{li}^g$	The ith task of the lth level of workflow G^g, $\bar{a}_{li}^g \in \bar{A}^g$; $\left\lvert \bar{a}_l^g \right\rvert$ is the number of tasks at lth level.
$SubD_i^g$	The sub-deadline of task a_i^g.
$\bar{t}^g$	The maximum completion time of the scheduled tasks of workflow G^g.
$\hat{t}$	The maximum completion time of all scheduled tasks.
η^g	The scheduled topology level of workflow G^g.
ρ_c^g	The pseudo-critical path length of workflow G^g.
μ_c^g	The urgency of workflow G^g.
$\bar{g}$	The workflow $G^{\bar{g}}$ to be scheduled.
α	The task to be scheduled, assuming it is the ith task of workflow $G^{\bar{g}}$, that is $\alpha = a_i^{\bar{g}}$.
$\tilde{t}_{\alpha h}^E$	The *Earliest Available Time*.
$\tilde{t}_{\alpha h}^A$	The *Actual Available Time*.
τ_i	The idle interval, $\tau_i = [\tau_{i0}, \tau_{i1}]$.
$\tilde{V}$	The available VM set.
$\hat{V}$	The intersection of the VM set with running tasks at the adjacent three levels ($\eta^{\bar{g}}$, $\eta^{\bar{g}} - 1$ and $\eta^{\bar{g}} - 2$) and the available VM set $\tilde{V}$.
$\bar{h}$	The instance selected to execute the task α.
π	The solution, $\pi = [\pi^L, \pi^A]$.
π^L	Encoding based on workflow topological levels in *Stage* 1.
π^A	Encoding in *Stage* 2; π^A is the vector description of $\bar{A}^g$.
π_{gli}^A	The ith task at the lth level of workflow G^g, $\pi_{gli}^A \in \pi^A$.
π^C	The continuous vector of π^L.

where $\bar{a}_{li}^g$ denotes the ith task at the lth level, $l = 1, 2, \ldots, \max\{L^g\}$, $i = 1, 2, \ldots, \left\lvert \bar{a}_l^g \right\rvert$, and $\left\lvert \bar{a}_l^g \right\rvert$ is the number of tasks at lth level. Especially, tasks at a lower topological level have higher priorities than tasks at a higher level (Wu et al., 2016).

4.1.3. Division sub-deadline

A task's latest finish (start) time is the latest time at which it can be finished (started) so that all tasks can be completed before deadline. In this paper, we set a sub-deadline for each task based on Latest Finish Time. We use the maximum execution time $\ddot{ET}_i^g = \frac{w_i^g}{\min_{p_k \in P}\{U(p_k)\}}$ and the maximum transmission time $\ddot{CT}_{ij}^g = \frac{d_{ij}^g}{\min_{p_k \in P}\{b_k\}}$ to calculate latest start time and finish time:

$$\ddot{FT}_i^g = \begin{cases} DL^g, & if\ Suc(a_i^g) = \emptyset, \\ \max\limits_{a_j \in Suc(a_i^g)} \left\{ \ddot{ST}_j^g - \ddot{CT}_{ji}^g \right\}, & otherwise. \end{cases}$$

$$\ddot{ST}_j^g = \ddot{FT}_j^g - \ddot{ET}_j^g.$$

The sub-deadline $SubD_i^g$ of task a_i^g is given in Eq. (21).

$$SubD_i^g = \begin{cases} \ddot{FT}_i^g, & if\ Suc(a_i^g) = \emptyset, \\ \varepsilon^g \times \ddot{FT}_i^g, & otherwise, \end{cases} \tag{21}$$

where ε^g is a random factor of the workflow G^g within the range $[0.95, 1]$. The smaller the ε^g, the more urgent the tasks in the workflow and the smaller the sub-deadline.

4.2. Heuristic scheduling algorithm based on 9 factors (HSA9Fs)

Heuristic scheduling algorithm is to dynamically determine the scheduling order of workflow topology level through the established rules. Each workflow is scheduled at the order of topology level. All tasks at the same level are independent of each other, without data transmission. They are scheduled according to the descending order of task weight. Algorithm 1 presents the procedure of HSA9Fs.

Algorithm 1: HSA9Fs

```
Input: Resource $P$, multi-workflow $G$
Output: Heuristic scheduling result
1  Set $\bar{t}^g$, $\hat{t}$ and $\eta_g$ to 0, where $G^g \in G$;
2  Set $V \leftarrow P$;
3  for $i \leftarrow 1$ to $\sum_{G^g \in G} |A^g|$ do
4      Compute $\rho_c^g$ by Eq. (22), $g \in G$, $c \in \{0,1\}$;
5      Compute $\mu_c^g$ by Eq. (23), $g \in G$, $c \in \{0,1\}$;
6      Get $\bar{g}$ through Eq. (24);
7      $\bar{l}^{\bar{g}} \leftarrow \bar{l}^{\bar{g}} + 1$;
8      Sort the tasks in $\bar{a}_{\bar{l}^{\bar{g}}}^{\bar{g}}$ in descending order of weight;
9      foreach $\alpha \in \bar{a}_{\bar{l}^{\bar{g}}}^{\bar{g}}$ do
10         call ResourceSelection;                    // Algorithm 2
11 call Adaptive adjustment solution;                 // Algorithm 3
```

4.2.1. Get the workflow to be scheduled

Multi-workflow scheduling first needs to select a workflow to schedule, which is determined by the pseudo-critical path and urgency of the workflow. They (pseudo-critical path and urgency) comprehensively consider factors such as unscheduled levels, current completion time and deadline of workflow.

Let $\bar{t}^g$ denote the maximum completion time of the scheduled tasks of workflow G^g, $\hat{t} = \max\left\{ \bar{t}^g \middle| G^g \in G \right\}$ denote the maximum completion time of all scheduled tasks, and $\bar{l}^g$ denote the scheduled topology level of workflow G^g.

The **pseudo-critical path**[2] is defined as the sum of the minimum execution time of tasks at each unscheduled level (that is, assume that tasks are executed on the instance with the highest processing capacity). In different cloud environments, the pseudo-critical path length are different. Let ρ_c^g denote the pseudo-critical path length of workflow G^g, $g \in G, c \in \{0,1\}$ (line 4 in Algorithm 1). If $c = 0$, it is in public cloud; otherwise, it is in private cloud.

$$\rho_c^g = \begin{cases} 0, & if\ \bar{l}^g = \max\{L^g\}, \\ \sum\limits_{l=\bar{l}^g+1}^{\max\{L^g\}} \dfrac{\max_{a_i^g \in \bar{a}_l^g}\left\{w_i^g\right\}}{\max_{p_k \in P^c}\left\{U(p_k)\right\}}, & otherwise. \end{cases} \tag{22}$$

The **urgency** of a workflow is the ratio of the pseudo-partial critical path to the remaining time (the difference between the deadline of the workflow and its current maximum completion time). Let μ_c^g denote the urgency of workflow G^g (line 5 in Algorithm 1). The urgency of a scheduled workflow is negative infinity ($-\infty$) by default.

$$\mu_c^g = \begin{cases} -\infty, & if\ \bar{l}^g = \max\{L^g\}, \\ \dfrac{\rho_c^g}{DL^g - \bar{t}^g}, & otherwise. \end{cases} \tag{23}$$

When $\mu_c^g, c = 0$ or 1 is in the range of $(0, 1)$, it means that the deadline can be met in the configuration with the highest instance execution ability; Otherwise, it will miss the deadline even in that configuration.

PMWS-HC should make the best use of private resources, so the workflow with the highest urgency based on private cloud ($c = 1$) will be the workflow to be scheduled, and the topology level of scheduling is the scheduled level plus 1, that is, $\bar{l}^{\bar{g}} = \bar{l}^{\bar{g}} + 1$. $\bar{g}$ (or $G^{\bar{g}}$) is the workflow to be scheduled (lines 6–7 in Algorithm 1).

$$\bar{g} = \arg\max_{G^g \in G} \mu_1^g. \tag{24}$$

[2] Since this critical path is based on the topology level, the tasks in the path are not necessarily on one path in the DAG, so it is called pseudo-critical path.

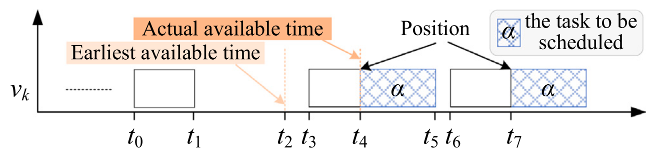


Fig. 4. Get the Earliest Available Time diagram.

4.2.2. *Task scheduling and VM selection*

After determining the workflow (or workflow topology level) to be scheduled, the tasks to be scheduled will be determined. In HSA9Fs, tasks are scheduled in descending order based on task weights at the same level (lines 8–10 in Algorithm 1). When selecting VM, we comprehensively consider the existing scheduling results, urgency, sub-deadline and other factors, as shown in Algorithm 2. Before resource selection, we first judge the privacy tag of the task to be scheduled. We only select resources from the private cloud if it is a sensitive task (lines 1–4 of Algorithm 2).

Let α be the task to be scheduled, assuming it is the ith task of workflow $G^{\bar g}$, that is $\alpha = a_i^{\bar g}$. The ***Earliest Available Time*** is the maximum value of the sum of the actual finish time of its predecessor task and transmission time. The Earliest Available Time $\tilde{t}^E_{\alpha h}$ of task α on VM v_h is (line 5 in Algorithm 2)

$$\tilde{t}^E_{\alpha h} = \begin{cases} Dur^C, & if\ \ Pre(\alpha)=\varnothing, \\ \max\limits_{a_j^{\bar g}\in Pre(\alpha)} \left\{FT_j^{\bar g} + CT_{j\alpha}^{\bar g,kh}\right\}, & otherwise, \end{cases} \tag{25}$$

where task $a_j^{\bar g}$ is executed on VM v_k, Dur^C is the duration of cold startup. The Earliest Available Time ignores the scheduled tasks on the VM and is used to calculate the Actual Available Time.

For Actual Available Time, we design a calculation method suitable for multi-workflow scheduling. According to the current scheduling result, we can get the idle interval $\tau_i = [\tau_{i0}, \tau_{i1}]$ on one VM, where $\tau_i \in \tau = \left\{\tau_i \middle| i = 1,2,\dots,|\tau|\right\}$. τ_{i0} and τ_{i1} denote the start time and end time of the ith interval τ_i, respectively. When constraint Eq. (26) is met, task α can be executed in the idle interval τ_i. The ***Actual Available Time*** $\tilde{t}^A_{\alpha h}$ depends on the idle interval that satisfies this constraint first (line 6 in Algorithm 2), as shown in Eq. (27).

$$\max\left\{\tilde{t}^E_{\alpha h}, \tau_{i0}\right\} + \frac{w_i^{\bar g}}{U(v_h^P)} \le \tau_{i1}, \tag{26}$$

$$\tilde{t}^A_{\alpha h} = \min\left\{\max\left\{\tilde{t}^E_{\alpha h}, \tau_{i0}\right\} | \tau_i \text{ satisfies Eq. (26)}, \tau_i \in \tau\right\}. \tag{27}$$

For example, α is the task to be scheduled, and Fig. 4 is a Gantt chart of a VM v_k. On this VM, the Earliest Available Time of task α is t_2, that is, it can be executed at any time after t_2 without affecting other tasks. The idle intervals after (including) t_2 are $[t_1, t_3]$, $[t_4, t_6]$ and $[t_7, +\infty]$, while the intervals $[t_4, t_6]$ and $[t_7, +\infty]$ can satisfy the execution time constraint Eq. (26). To execute task α as early as possible, we take t_4 as its Actual Available Time on this VM.

When the Actual Available Time plus the task's execution time does not exceed the sub-deadline of the task, the VM is deemed to meet the sub-deadline, and all such VMs constitute an available VM set $\tilde{V}$. We select the VM to execute the task through the following three-layers specific rules:

- Lines 8-9 in Algorithm 2. In the hybrid cloud, when the urgency $\mu^{\bar g}$ is not within $(0, 1)$, or when the available VM set $\tilde{V}$ is empty, the VM is selected according to the earliest completion time, as shown in Eq. (28).

$$\bar{h} = \arg\min_{h\in V}\left\{\tilde{t}^A_{\alpha h} + \frac{w_i^{\bar g}}{U(v_h^P)}\right\}. \tag{28}$$

Algorithm 2: ResourceSelection

```
    Input: Workflow $\bar g$ ($G^{\bar g}$), task $\alpha$ (or $a_i^{\bar g}$), VMs set $V$
    Output: Scheduling result of the task
 1  if $\phi_i^{\bar g} == 1$ then                    // Sensitivity judgment
 2      $V' \leftarrow$ the VMs of the private cloud in $V$;
 3  else
 4      $V' \leftarrow V$;
 5  Compute Earliest Available Time $\tilde{t}^E_{\alpha h}$ by Eq. (25), $h \in V'$;
 6  Compute Actual Available Time $\tilde{t}^A_{\alpha h}$ by Eqs. (26) and (27),
      $h \in V'$;
 7  Get VM set $\tilde{V}$ that meets sub-deadline through $\tilde{t}^A_{\alpha h}$;
 8  if $\mu^{\bar g} \notin (0,1)$ or $\tilde{V} == \varnothing$ then
 9      Get $\bar h$ by Eq. (28), $\bar h \in V'$;
10  else
11      Get the intersection $\hat{V}$ of VM set with running tasks at the
        adjacent three levels ($\bar l^{\bar g}$, $\bar l^{\bar g} - 1$ and $\bar l^{\bar g} - 2$) and the VM set $\tilde{V}$;
12      if $\hat{V} \neq \varnothing$ then
13          Get $\bar h$ by Eq. (29a), $\bar h \in V'$;
14          if $\bar h$ is not satisfied with Eq. (29b) then
15              Get $\bar h$ by Eq. (30), $\bar h \in V'$;
16  $\zeta^{\bar g}_{i\bar h} \leftarrow 1$, $ST_i^{\bar g} \leftarrow \tilde{t}^A_{\alpha\bar h}$, $FT_i^{\bar g} \leftarrow ST_i^{\bar g} + \frac{w_i^{\bar g}}{U(v_{\bar h}^P)}$;
17  $\hat{t}^{\bar g} \leftarrow \max\{\hat{t}^{\bar g}, FT_i^{\bar g}\}$, $\hat{t} \leftarrow \max\{\hat{t}, \hat{t}^{\bar g}\}$ ;
18  if only task $\alpha$ is running on VM $\bar h$ then $V \leftarrow V \cup [v_{\bar h}^P]$;
```

- Lines 11-13 in Algorithm 2. For compact scheduling, the VM is selected from the following set: the intersection $\hat{V}$ of the VM set with running tasks at the adjacent three topological levels ($\bar l^{\bar g}$, $\bar l^{\bar g} - 1$ and $\bar l^{\bar g} - 2$) and the available VM set $\tilde{V}$. In this intersection $\hat{V}$, in order to preferentially select one VM in the private cloud, we select the corresponding VM using the minimum correction parameter (Actual Available Time multiplied by $(1 - 0.9\eta_h)$), as shown in Eq. (29a). When the finish time of task α on this VM does not exceed the sub-deadline $SubD_i^{\bar g}$ or the current maximum completion time $\hat{t}$ (as shown in Eq. (29b)), the VM is used to execute this task.

$$\begin{cases} \bar h = \arg\min\limits_{h\in\hat V}\left\{\tilde{t}^A_{\alpha h} \times (1 - 0.9\eta_h)\right\}, & (a) \\ \tilde{t}^A_{\alpha\bar h} + \dfrac{w_i^{\bar g}}{U(v_{\bar h}^P)} \le \min\left\{SubD_i^{\bar g}, \hat{t}\right\}. & (b) \end{cases} \tag{29}$$

- Lines 14-15 in Algorithm 2. In the VM set $\tilde{V}$, the corresponding VM is selected according to the another correction parameter, as shown in Eq. (30).

$$\bar h = \arg\min_{h\in\tilde V}\left\{\left(DL^{\bar g} - \tilde{t}^A_{\alpha h} - \frac{w_i^{\bar g}}{U(v_h^P)}\right) \times \frac{w_i^{\bar g}}{U(v_h^P)} \times \frac{1+\eta_h}{1+\tilde{t}^D_{\alpha h}}\right\}, \tag{30}$$

where $\left(DL^{\bar g} - \tilde{t}^A_{\alpha h} - \frac{w_i^{\bar g}}{U(v_h^P)}\right)$ is the remaining available time before the deadline and $\frac{w_i^{\bar g}}{U(v_h^P)}$ is the execution time of the current task. The product of these two parameters is to select a VM whose Actual Available Time is not too early or too late, which makes the algorithm non-greedy. $1 + \eta_h$ is the preferred private cloud. $\tilde{t}^D_{\alpha h} = \tilde{t}^A_{\alpha h} - \tilde{t}^E_{\alpha h}$ is the difference between the Actual Available Time and the Earliest Available Time, plus 1 to avoid the denominator being 0.

When the VM $\bar h$ that executes the task α is obtained, the actual start time of this task is also determined, e.g. $ST_i^{\bar g} = \tilde{t}^A_{\alpha\bar h}$. Meanwhile, its actual finish time can also be obtained (line 16 in Algorithm 2). If only task α is executed on VM $\bar h$, the VM $\bar h$ will be added to the used VM instance set V (line 18 in Algorithm 2). Particularly, when the number of this type of VM has reached the limit of private cloud resources, it will not be added.

4.2.3. Adaptive calibration task actual start time

While designing scheduling algorithm, there is an universal commonality: tasks are executed as early as possible. There will be some idle time caused if subsequent tasks cannot instantly start to be executed in time owing to dependency constraints. However, some idle time can be avoided by delaying the start execution time of some tasks.

Definition 3 (*Block Structure* Sun et al., 2022**).** It consists of tasks that are continuously executed without idle intervals on the same VM. Particularly, when there is only one task, it can also be called a block structure.

Theorem 1 (*Block Structure Property* Sun et al., 2022**).** *In a given scheduling solution, first block structure on VM v_h is $X = [1, 2, \dots, |X|]$. When $\forall x \in X, \hat{t}^F_x - t^F_x > 0$, the block structure can be moved backward by Δt without affecting the execution of other tasks to reduce idle time and cost.*

$$\hat{t}^F_x = \begin{cases} \min\left\{ST_y - CT^{v_h,v_k}_{x,y}\right\}, & y \in \left(Suc(x) - Suc(x) \cap X\right), \\ FT_x, & if\ \ Suc(x) = \emptyset, \end{cases} \tag{31}$$

$$\Delta t = \min\left\{ST_{|X|+1} - FT_{|X|}, \min\left\{\hat{t}^F_x - FT_x | x \in X\right\}\right\}, \tag{32}$$

where $\hat{t}^F_x$, ST_x and FT_x denote the estimated latest finish time, actual start time and actual finish time of task x in the current solution.

Algorithm 3: Adaptive adjustment solution

```
Input: The mapping between tasks and resources $\zeta$, the actual
       start of tasks $ST$, and the actual finish time of tasks $FT$.
Output: $f = (\mathbb{T}, \mathbb{E}, \mathbb{M})$
/* Calibrate task Actual Start Time */
1  Procedure CalibrateTime($X$)
2      Compute $\hat{t}^F_x$ using Eq. (31), $\forall x \in X$;
3      if $\forall x \in X$, $\hat{t}^F_x > FT_x$ then
4          Compute $\Delta t$ using Eq. (32);
5          foreach $x \in X$ do
6              $ST_x \leftarrow ST_x + \Delta t$;
7              $FT_x \leftarrow FT_x + \Delta t$;
8  repeat
9      foreach $h \in V$ do
10         Find $X$ as first block structure in instance $v_h$;
11         CalibrateTime($X$);
12         Find another $X$ that ends with the penultimate task in
            instance $v_h$;
13         CalibrateTime($X$);
14 until no block is found;
/* Setting VM hibernation */
15 foreach $h \in V$ do
16     Record tasks $S_{hj}$ on instance $v_h$;
17     $var \leftarrow 0$, $\hbar \leftarrow 1$, $\hbar' \leftarrow 0$;
18     $\overline{ST}_{h\hbar} \leftarrow ST_{S_{h1}} - Dur^C$;
19     for $k \leftarrow 1$ to $|S_h| - 1$ do
20         $p \leftarrow S_{hk}, s \leftarrow S_{h(k+1)}$;
21         if $ST_s - FT_p > Dur^H$ & $FT_p - var > Gap^H$ then
22             $\hbar' \leftarrow \hbar' + 1$, $var \leftarrow ST_s$;
23             $\overline{ST}^H_{h\hbar'} \leftarrow FT_p$, $\overline{FT}^H_{h\hbar'} \leftarrow ST_s - Dur^W$;
24             $\overline{FT}_{h\hbar} \leftarrow FT_p$, $\hbar \leftarrow \hbar + 1$, $\overline{ST}_{h\hbar} \leftarrow ST_s$;
25     $\overline{FT}_{h\hbar} \leftarrow FT_{S_{h|S_h|}}$;
26 Compute $f = (\mathbb{T}, \mathbb{E}, \mathbb{M})$.
```

Table 4
Solution representation in *Stage* 1.

Workflow G^g	1		2			3			
$\max(L^g)$	2		3			4			
Dimension l	1	2	3	4	5	6	7	8	9
π^C_l	2.25	1.69	4.35	1.57	0.64	4.83	3.56	2.68	0.93
y_l	5	4	8	3	1	9	7	6	2
π^L_l	2	2	3	2	1	3	3	3	1

Delaying task execution to save cost reflects the advantages of the cloud's pay-as-you-go. Consequently, the scheduling solution is adjusted utilizing Theorem 1: Traverse two block structures on each VM until there is no block structure that satisfies the constraint of Theorem 1 (lines 8–14 in Algorithm 3). If the block structure satisfies Theorem 1's constraint, the actual start and finish time of corresponding tasks will be updated. Delaying the operation will not lead to the increase of the maximum completion time of the workflow, because it limits the estimated latest completion time of tasks without successors to the actual completion time. Therefore, the deadline will not be missed.

4.2.4. Adaptive setting VM hibernation

Since certain instances support the hibernation function, if one instance has been idle for an extended period of time, it is prudent to hibernate the instance to save cost. However, excessive hibernation can cause system failure or software function errors. As a result, a heuristic strategy for scheduling when to hibernate an instance is proposed: Traverse each idle interval between in instances, and set the idle interval to hibernate state when the hibernation criteria is reached. The hibernation criteria may be the shortest duration of hibernation and the minimum gap between two adjacent hibernation in one instance. According to the scheduled results, the tasks executed on each instance and their start and finish time are known. Let S_{hj} be the jth task executed on instance v_h. See lines 15–25 Algorithm 3 for details.

4.3. Meta-heuristic algorithm

4.3.1. Encoding and population initialization

Encoding. The encoding of algorithm is to determine the scheduling order of workflows and their tasks. Because of the continuous nature of MSSA, it cannot be directly applied to multi-workflow scheduling. Therefore, it is crucial to develop an appropriate mapping scheme to transform individuals (salps, continuous vectors) and task sequences (discrete vector). In this paper, we schedule each DAG in the order of topological level, and present the following two-stage encoding method.

- *Stage* 1. Encoding based on workflow topological levels, the encoding length is the sum of the number of topological levels of all workflows. The workflow index and its times are used to indicate the tasks of the workflow topological level that will be scheduled, e.g. $\pi^L = \left[\pi^L_1, \pi^L_2, \dots, \pi^L_9\right] = \left[2, 2, 3, \underline{2}, 1, \underline{3}, 3, 3, 1\right]$, where $\pi^L_4 = 2$ denotes the tasks at 3-rd level of workflow G^2; $\pi^L_6 = 3$ denotes the tasks at 2-nd level of workflow G^3.
- *Stage* 2. Tasks within the topological level corresponding to *Stage* 1, which are independent of each other and have no data transfer. Let π^A_{gli} be the ith task in the lth level of workflow G^g (π^A is the vector representation of $\bar{A}^g$ in Section 4.1.2).

Therefore, the whole solution is expressed as $\pi = [\pi^L, \pi^A]$.

Smallest-order-value (SOV) rule (Qian et al., 2011; Sun et al., 2018) are applied to conversions between continuous and discrete individuals,[3] i.e. $\pi^C \Leftrightarrow \pi^L$. In the SOV rule, continuous vectors $\pi^C =$

[3] Note: MSSA acts on *Stage* 1; IGA acts on both *Stage* 1 and *Stage* 2. Therefore, there is continuous coding in *Stage* 1 only.

$\left[\pi_1^C, \pi_2^C, \dots, \pi_{|L|}^C\right]$ are ranked in ascending order to get an sequence $y = \left[y_1, y_2, \dots, y_{|L|}\right]$, where y_i denotes the index of original π_i^C in the sorted π^C, $|L| = \sum_{g\in G} \max(L^g)$ is the encoding length. Then fill π^L with $\max(L^g)$ times g in the order of y values in turn. To better understand the SOV rule, an example is provided in Table 4. In the example, $|G| = 3, |L| = \sum_{g\in G} \max(L^g) = 2+3+4 = 9$. In the ascending order of π^C,

- when π_5^C is the 1-st, then $y_5 = 1$ and $\pi_5^L = 1$;
- when π_9^C is the 2-nd, then $y_9 = 2$ and $\pi_9^L = 1$;
- when π_4^C is the 3-rd, then $y_4 = 3$ and $\pi_4^L = 2$;
- when π_2^C is the 4-th, then $y_2 = 4$ and $\pi_2^L = 2$, and so on.

Population Initialization. An initial solution is obtained by HSA9Fs (Section 4.2). Others are randomly generated: In *Stage* 1, continuous vectors are randomly generated, and then discrete vectors are generated by SOV rule; *Stage* 2 is the random permutation of tasks at the same topological level; Then the Algorithm 2 in Section 4.2.2 is utilized to **decode**.

4.3.2. Update mechanism of MSSA

Multi-objective Salp Swarm Algorithm (MSSA) is proposed by Mirjalili et al. (2017) for optimization problems on continuous domains. By mimicking the swarming behavior of salps, MSSA models the salp chains as two groups: leader and followers. The leader is the salp at the front of the chain, while followers are the rest of the salps. The salp leader guides the swarm, as the name implies, while the followers follow one by one.

Let x_j^i be the position of ith salp in the jth dimension, where $i = 1, 2, \dots, popsize$; $j = 1, 2, \dots, |L|$. Each salp can be updated by Eq. (33) or Eq. (34).

$$x_j^i = \begin{cases} F_j + c_1 \times \left((ub_j - lb_j) \times c_2 + lb_j\right), & if\ c_3 \le 0.5, \\ F_j - c_1 \times \left((ub_j - lb_j) \times c_2 + lb_j\right), & otherwise, \end{cases} \tag{33}$$

$$x_j^i = \frac{1}{2}(x_j^i + x_j^{i-1}), \tag{34}$$

where F_j is the position of source food in jth dimension, ub_j and lb_j denote the limits of search domain at dimension j, c_1, c_2 and c_3 are random values follow the uniform distribution $U[0,1)$. Since Eq. (33) only updates its position in relation to F_j, to balances exploration and exploitation, c_1 is redefined as $c_1 = 2\exp\left(-(\frac{4IR}{|IR|})^2\right)$, where IR and $|IR|$ denote the current iteration and the total number of iterations.

Source food F is the current optimal individual or a individual in the archive set AS. This AS retains the best non-dominated solutions discovered thus far during iteration and has a maximum size $|AS|$ to store a limited number of non-dominated solutions. During iteration, each salp is compared with all solutions in AS by using the Pareto domination definition. If it can dominate a solution in AS, it will replace the dominated solution; If it cannot dominate all solutions, the salp will be added to the AS.

When the number of solutions in AS exceeds $|AS|$, we invoke the method of Mirjalili et al. (2017) to remove the redundant solutions from the dense neighborhood's non-dominated solutions.

- The maximum distance is $\bar{d} = [\bar{d}_1, \bar{d}_2, \bar{d}_3] = \left[\frac{\mathbb{T}_{max} - \mathbb{T}_{min}}{|AS|}, \frac{\mathbb{E}_{max} - \mathbb{E}_{min}}{|AS|}, \frac{\mathbb{M}_{max} - \mathbb{M}_{min}}{|AS|}\right]$, where $\mathbb{T}_{max}$, $\mathbb{T}_{min}$, $\mathbb{E}_{max}$, $\mathbb{E}_{min}$, $\mathbb{M}_{max}$, and $\mathbb{M}_{min}$ are the maximum and minimum values of the corresponding objectives in AS, respectively.
- The density of a solution is the number of solutions within the maximum distance $\bar{d}$ around it.

According to the density, we utilize the Roulette Wheel Strategy (Wu & Wang, 2018) to eliminate the redundant solution, and select an individual as the Source food F in the same way.

Algorithm 4: MSIA

```
   Input: Resource $P$, multi-workflow $G$
   Output: Archive set AS
   /* Initialize population                                  */
 1 Initialize the population with HSA9Fs and random method;
    // Algorithms 1, 2 and 3
   /* Algorithm optimization                                 */
 2 for $IR \leftarrow 1$ to $|IR|$ do
       /* MSSA                                               */
 3     Initialize or update the archive set AS;
 4     $c_1 \leftarrow 2\exp\left(-(\frac{4IR}{|IR|})^2\right)$;
 5     Choose a Source food $F$ from the archive set AS;
 6     for $i \leftarrow 1$ to $popsize$ do
 7         if $i \le popsize/2$ then
 8             Update the salp $x^i$ by Eq. (33);
 9         else
10             Update the salp $x^i$ by Eq. (34);
11         Shuffle Stage 2 randomly;
12         Let $\pi^C \leftarrow x^i$, $\pi^L \Leftrightarrow \pi^C$ ;              // SOV rule
13         Calculate the fitness of $\pi = [\pi^L, \pi^A]$ by Algorithms 2
            and 3;
           /* IGA                                            */
14         Let $\bar{\pi} \leftarrow \pi$;
15         for $IR' \leftarrow 1$ to 10 do
16             $\pi', \pi'' \leftarrow Destruction(\bar{\pi})$;
17             $\bar{\pi} \leftarrow Reconstruction(\pi', \pi'')$;
18             Calculate the fitness of $\bar{\pi}$ by Algorithms 2 and 3;
19             if $\bar{\pi}$ dominate $\pi$ then
20                 Let $\bar{\pi}^L \Leftrightarrow \bar{\pi}^C$, $\pi \leftarrow \bar{\pi}$ ;         // SOV rule
21                 break;
22 Update and output the archive set AS;
```

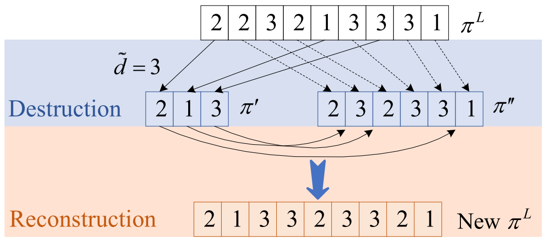


Fig. 5. Illustration of destruction and reconstruction.

4.3.3. Update mechanism of IGA

The Iterative Greedy Algorithm (IGA) was proposed by Ruiz and Stützle (2007) for flowshop scheduling problem to minimize makespan. The incumbent solution is partially destroyed at each iteration of IGA, and the removed elements are greedily reinserted back into the solution to build a new solution. These two operators are destruction and reconstruction, respectively. If the new solution can dominate the incumbent solution, the incumbent solution will be replaced. Illustration of destruction and reconstruction is shown in Fig. 5.

- **Destruction**. Randomly extract $\tilde{d}$ elements from a given sequence π^L of *Stage* 1 (or π_{gl}^A of *Stage* 2: tasks of lth level in workflow G^g) and get two sub-sequences π' and π''. π' are composed of the extracted elements and π'' is composed of the remaining elements. $\tilde{d}$ is destruction parameter. For π^L of *Stage* 1, $\tilde{d}$ is generated randomly from a discrete uniform distribution $U\left[\left\lceil\frac{|L|}{|G|}\right\rceil, 2 \times \left\lceil\frac{|L|}{|G|}\right\rceil\right]$. For each level π_{gl}^A of *Stage* 2, $\tilde{d}$ is generated randomly from a discrete uniform distribution $U\left[1, \left\lceil\frac{1}{2} \times \left|\pi_{gl}^A\right|\right\rceil\right]$.

Table 5
Configurations and prices of VMs in public cloud.

VM Type	Processing Capacity (GFLOPS)	On-Demand Cost ($/h)	Bandwidth (Gbps)[a]	Transmission Cost ($/Gb)
1 c3.large	30.8	0.128	1	
2 c3.xlarge	61.6	0.255	1.5	
3 c3.2xlarge	123.2	0.511	2	0.02
4 c3.4xlarge	242	1.021	3	
5 c3.8xlarge	475.2	2.043	3	

[a]Manually set according to the VM configuration.

Table 6
Configurations and powers of VMs in private cloud.

VM Type (Numbers)	Processing Capacity (GFLOPS)	Bandwidth (Gbps)	Dynamic Power (W)	Idle Power (W)	Transmission Power (W)
1 (3)	44	1.25	110	10	
2 (3)	66	1.75	190	20	5
3 (4)	96.8	2.5	300	35	

- **Reconstruction**. All extracted elements (π') are randomly reinserted one by one into possible positions of π'' to form a new complete sequence.

Consequently, in Algorithm 4, we detail the pseudo-code of our proposed MSIA. Line 1 is the initialization population. Lines 3–13 are to update the population using MSSA. Lines 14–21 are to further optimize individual using IGA. The **relationship** between HSA9Fs and MSSA is given as follows:

- MSSA is a swarm intelligence optimization algorithm that can get more non-dominated solutions through population iteration. We use the solution generated by HSA9Fs as an initial individual of MSSA to guide the algorithm evolution.
- The encoding of MSSA is the execution sequence of workflow and its tasks, while its decoding is based on the Resource Selection part of HSA9Fs.

Our HSA9Fs is a heuristic algorithm, which can only obtain one solution. However, the solution of HSA9Fs lacks the diversity of multi-objective optimization, so we employ MSSA to pursue this purpose. HSA9Fs can intelligently select the task to be scheduled and the VM to execute that task according to the scheduling situation of multiple workflows and the current state of the hybrid cloud system. Therefore, HSA9Fs has good coverage in the scheduling process (as proven by the performance results in Section 5.3).

5. Performance evaluation and results

5.1. Simulation environment setup

5.1.1. Resource environment

For public cloud, we use 5 representative VM types from low to high configuration, as shown in Table 5. The VM configurations and their processing capacity are based on current Amazon EC2 platform.[4] The cold startup time Dur^C, the warm startup time Dur^W and the duration of stopping Dur^P are set to 55.9s, 34.0s and 5.6s, respectively (Sun et al., 2022). The shortest duration of hibernation Dur^H is 60.0s. The minimum gap between two adjacent hibernation in one instance Gap^H is 120s. Unit price of VM in hibernation state M^H (only charge for ElasticIP cost) is 0.005$/h.

[4] https://aws.amazon.com/ec2/pricing/on-demand/, https://instances.vantage.sh/

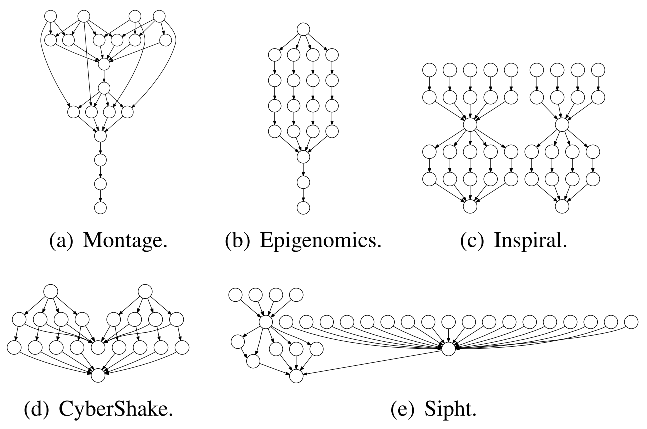


Fig. 6. Five real-world workflow applications.

For private cloud, we use 3 representative VM types from low to high configuration, as presented in Table 6. Assume that there are 10 VMs of three types, with 3, 3 and 4 VMs of each type. The VMs are cold started before executing tasks. Energy consumption is calculated only for VMs with tasks.

5.1.2. Workflow applications

Five real-world workflow applications with different scales from different scientific areas are adopted in the simulation (Juve et al., 2013; Rodriguez & Buyya, 2014; Sahni & Vidyarthi, 2018), as shown in Fig. 6. All these workflows are generated in the form of Directed Acyclic Graph in XML (DAX) format by Pegasus WorkflowGenerator (Juve et al., 2013; Sahni & Vidyarthi, 2018), and are publicly available on Pegasus website.[5] These DAX files contain information such as list of tasks and dependencies between tasks. More details about these workflows can be found in Juve et al. (2013).

Each workflow type has 4 scales. Fig. 6(a) Montage has 25, 50, 100, 1000 tasks; Fig. 6(b) Epigenomics has 24, 64, 100 and 997 tasks; Fig. 6(c) Inspiral has 30, 50, 100 and 1000 tasks; Fig. 6(d) CyberShake has 30, 50, 100 and 1000 tasks; Fig. 6(e) Sipht has 30, 60, 100 and 1000 tasks. Each workflow has 4 deadline factors from loose to tight. Under each deadline factor, the privacy tag of task is set with a probability of 20%, and it is also generated 4 times. Therefore, the total number of workflows is $5 \times 4 \times 4 \times 4 = 320$.

We randomly choose the different number of workflows as test problems to simulate the multi-workflows submitted by users, which are denoted by $|G|\_\sum|A^g|$. The number of workflows $|G| \in \{3,5,7,10\}$ and the total number of tasks $\sum|A^g| < 1000$ constitute Medium-Small-Scale test problems; the number of workflows $|G| \in \{5,15,25\}$ and the total number of tasks $\sum|A^g| \geq 1000$ constitute Large-Scale test problems. There are $4 \times 2 + 3 \times 3 = 17$ groups of test problems.

The following rule (Sahni & Vidyarthi, 2018; Wu et al., 2016) is used to assign a deadline for workflow. We use the maximum execution time $\widehat{ET}_i^g = \max\left\{\frac{w_i^g}{U(v_h^P)}\middle| p_h \in P^1\right\}$ and the maximum transmission time $\widehat{CT}_{ij}^g = \max\left\{\frac{d_{ij}^g}{b_h}\middle| p_h \in P^1\right\}$ to estimate start time and finish time:

$$\widetilde{ST}_i^g = \begin{cases} 0, & if\ \ Pre(a_i) = \emptyset, \\ \max_{a_j \in Pre(a_i)} \left\{ \widetilde{FT}_j^g + \widehat{CT}_{ji}^g \right\}, & otherwise. \end{cases}$$

$$\widetilde{FT}_j^g = \widetilde{ST}_j^g + \widehat{ET}_j^g.$$

[5] https://confluence.pegasus.isi.edu/display/pegasus/Deprecated+Workflow+Generator

The deadline is set to:

$$DL^g = \lambda \times \max \left\{ \widetilde{FT}_i^g \middle| a_i^g \in A^g \right\}, \tag{35}$$

where $\lambda \in \{0.8, 1.1, 1.5, 1.8\}$ is deadline factor.

5.2. Baseline algorithms and evaluation metrics

To evaluate the performance of the proposed algorithms, three state-of-the-art and related multi-objective workflow scheduling algorithms are implemented for comparison, including MACO (Li et al., 2019), GMPSO (Hafsi et al., 2022), GALCS (Xia et al., 2022). The parameter configurations of the compared algorithms are all based on the suggestions in the corresponding references.

- MACO mainly relies on pheromone matrices and heuristic probability matrices to generate and update the scheduling solutions. In MACO, $\alpha = 1$, $\beta = 10$, $\rho = 0.2$, $Q_1 = 10$, $Q_2 = 0.1$, and $populationsize = 100$.
- GMPSO is formed by incorporating NSGAII into MOPSO and applies a novel solution encoding that represents the task ordering, the task mapping and the resource provisioning processes of the workflow scheduling problem in hybrid Clouds. In GMPSO, $LeadersSize = 50$, $CrossoverProbability = MutationProbability = 0.5$, $GMParents = 50$, $GMJump = 3$, $populationsize = 100$, $C1, C2 \in [1.5, 2.5]$ and $r1, r2 \in [0, 1]$.
- GALCS is combined GA with the longest common subsequence algorithm, aiming to record the favorable gene blocks and to improve the performance of GA. In GALCS, $\delta = populationsize = 30$.

In addition to the above algorithms, we also consider the performance comparison with two variant algorithms of MSSA and HSA9Fs.

- MSSA1 is encoded and decoded in the same way as MSIA, but its population is generated randomly during initialization, i.e., the solution of HSA9Fs is not used to guide the algorithm search.
- MSSA2 is encoded in the same way as MSIA, but decoded using the following approach: according to the rule of the earliest finish time (the task will be executed on whichever instance it is finished the earliest), the private cloud resources will be used first. For non-sensitive tasks, only when their execution in the private cloud cannot meet the (sub-)deadline will they lease resources from the public cloud.
- HSA9Fs is essentially a heuristic algorithm, which obtains solutions as feasible solutions. Compared with HSA9Fs, we can evaluate the performance of our heuristic algorithm and the quality of the initial solution of MSIA.

Neither MSSA1 nor MSSA2 adopts Algorithm 3 to adapt to the calibration task, and their comparison can evaluate the performance of our proposed Resource Selection Algorithm 2. Algorithm 2 is also the decoding method of MSIA. We can evaluate the performance of HSA9Fs and Algorithm 3 (adaptive calibration task) by comparing them (MSSA1 and MSSA2) with MSIA. In MSIA, MSSA1 and MSSA2, $populationsize = 30$, the total number of iterations $|IR| = 100$ and the maximum number of non-dominated solutions $|AS| = 20$.

We quote the popular and effective method in Li et al. (2022), Pan et al. (2021) (merging multiple workflows into one large workflow by adding two virtual nodes), so that algorithms MACO and GMPSO can solve multiple workflows. To mitigate the effects of experimental uncertainty, each algorithm is repeated 10 times. For different test problems, the given running time of each algorithm is $1.2 \times \sum |A^g|$ seconds. All the simulations are conducted in Python and are executed on a 64 bit PC with Intel Core i5-9500 3.0 GHz, 32 GB RAM and Ubuntu 20.04.2 LTS. The source codes are publicly available at https://doi.org/10.24433/CO.5717787.v2 and https://github.com/zaixing-sun/MSIA_PMWS_HC_Public.

In this paper, we have two evaluation metrics to measure the performance of algorithms as follows:

(a) **HyperVolume (HV)** (Chen et al., 2019; Li et al., 2022; Pan et al., 2021; Saeedi et al., 2020). HV evaluates the diversity and convergence of an evolutionary algorithm. It is obtained by calculating the volume of the enclosed area between a set of solutions generated by the algorithm and a reference point, which is usually selected as the maximum objective values, e.g., the highest Total Tardiness, Total Energy Consumption and Total Monetary Cost. Specifically, we utilize Relative Percentage Deviation (RPD, Eq. (36)) to normalize three objectives values of solutions. The reference point $(1.0, 1.0, 1.0)$ can, thus, be used for calculating HV. Hence, the range of HV is $[0, 1]$ and a larger HV value is preferable, which indicates that the obtained set of nondominated solutions is closer to the Pareto front and has a desired distribution.

$$RPD^A = \frac{f^A - f_{min}}{f_{max} - f_{min}}, \tag{36}$$

where f^A is the solution obtained by algorithm A, and f_{min} and f_{max} are the minimum and maximum values achieved among all algorithms, respectively.

(b) **Coverage Rate (CR)** (Chen et al., 2019; Pan et al., 2021; Wu & Wang, 2018). This metric is used to compare the archive sets AS_1 and AS_2, which reflects the dominance relationship between the solutions in the two sets. CR is formulated as Eq. (37).

$$CR(AS_1, AS_2) = \frac{\left| \left\{ x_2 \in AS_2 \middle| \exists x_1 \in AS_1 : x_2 \prec x_1 \right\} \right|}{|AS_2|}, \tag{37}$$

where $CR(AS_1, AS_2)$ represents the proportion of solutions in AS_2 that are dominated by the solutions in AS_1. $CR(AS_1, AS_2) > CR(AS_2, AS_1)$, indicates that AS_1 is better than AS_2 in terms of convergence under the Pareto Optimal metric. In addition, the value of CR is between 0 and 1, and $CR(AS_1, AS_2) + CR(AS_2, AS_1) \neq 1$.

5.3. Performance results

We analyze these results in terms of HyperVolume, Coverage Rate, running time and number of leased VMs in public cloud. The results are the mean of 10 runs of algorithms.

5.3.1. Comparison of HyperVolume

Table 7 shows the comparison of HyperVolume by different algorithms. According to the results of HV, MSIA is superior to other algorithms in 68.42%(13/19) cases. For Medium-Small-Scale, the HV criterion of MSIA has improved up to 2.13%, 6.42%, 36.80%, 38.14%, 49.13%, and 74.37% respectively, compared to MSSA1, MSSA2, GMPSO, HSA9Fs, GALCS, and MACO algorithms. For Large-Scale, the HV criterion of MSIA has improved up to 1.31%, 15.17%, 19.74%, 26.15%, and 53.18% respectively, compared to MSSA1, GMPSO, HSA9Fs, MSSA2, and GALCS algorithms. The superiority of MSSA1 over MSSA2 proves the efficiency of Algorithm 2: ResourceSelection, which is particularly evident on Large-Scale. GMPSO outperforms on Large-Scale than on Medium-Small-Scale. The reasons for the poor performance of MACO are: (i) it relies on probability selection and spends a lot of effort on updating pheromone matrix and probability matrix; (ii) it is based on the traditional encoding method, and each individual must repair the violation of task execution constraints. These reasons make it difficult for MACO to obtain feasible solutions on Large-Scale within a given time. In addition, based on HSA9Fs, the HV of MSIA is significantly improved, which indicates MSIA further searches in the global (MSSA) and local (IGA), yielding results with excellent diversity and convergence.

To capture the performance difference between the algorithms intuitively, we make a one-way ANOVA of HV results. Fig. 7 shows the mean plot of HV with 95.0 percent Tukey HSD confidence intervals for all algorithms. All p-values are less than 0.05 for Medium-Small-Scale and Large-Scale test problems, indicating that these algorithms

Table 7
The mean of HyperVolume of different algorithms[a].

	$\lvert G\rvert\_\sum\lvert A^g\rvert$	MACO[b]	GMPSO	GALCS	MSSA1	MSSA2	HSA9Fs	MSIA
Medium-Small-Scale	3_180	0.095	0.530	0.340	**0.808**	**0.974**	0.693	0.803
	3_200	0.142	0.542	0.395	**0.982**	0.850	0.055	**0.985**
	5_330	0.104	0.466	0.543	0.884	**0.925**	0.727	**0.900**
	5_430	0.086	0.470	0.389	**0.911**	0.752	0.757	**0.952**
	7_391	0.041	0.380	0.236	**0.528**	0.506	0.212	**0.577**
	7_394	0.075	0.397	0.306	**0.611**	0.418	0.392	**0.623**
	10_649	0.001	0.381	0.150	**0.877**	0.820	0.282	**0.904**
	10_672	0.002	0.384	0.205	0.722	**0.733**	0.325	**0.750**
	Avg[c]	0.068	0.444	0.320	**0.790**	0.748	0.430	**0.812**
Large-Scale	5_1230	\[b]	0.512	0.266	**0.654**	**0.671**	0.366	0.592
	5_1300	\	0.436	0.212	**0.837**	0.797	0.569	**0.884**
	5_3150	\	**0.497**	0.328	0.445	0.333	0.326	**0.462**
	15_2656	\	0.486	0.135	**0.860**	0.542	0.445	**0.852**
	15_4681	\	0.564	0.094	**0.727**	0.350	0.570	**0.768**
	15_5585	\	0.702	0.154	**0.769**	0.366	0.680	**0.811**
	25_6164	\	0.573	0.118	**0.615**	0.309	0.523	**0.631**
	25_7984	\	0.505	0.119	**0.698**	0.242	0.609	**0.706**
	25_10878	\	**0.665**	0.092	0.582	0.343	0.439	**0.598**
	Avg[c]	\	0.549	0.169	**0.688**	0.439	0.503	**0.701**

[a]Best results are in bold and underlined, and second-best results are in bold.
[b]MACO cannot output feasible solutions in a given time for large-scale test problems. Therefore, it is indicated by '\'.
[c]*Avg* is the average value of the algorithm based on corresponding scale problems.

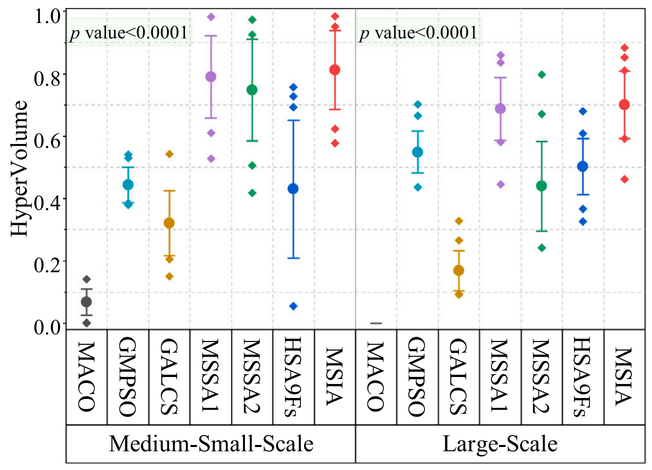
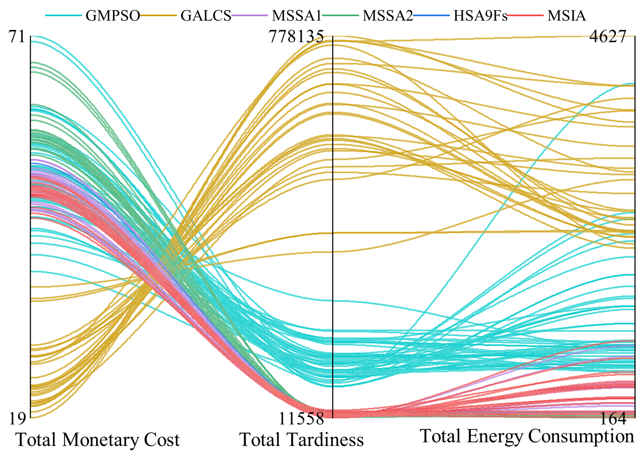


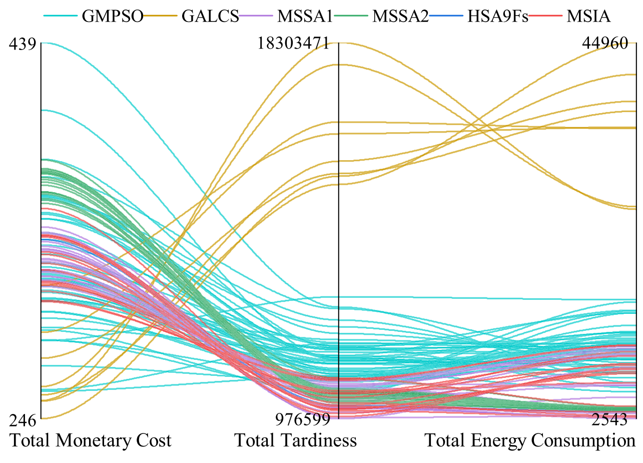
Fig. 7. The mean plot of HV with 95.0 percent Tukey HSD confidence intervals.

are statistically significant different at the 95.0% confidence level. For Medium-Small-Scale, there is no significant difference among MSIA, MSSA1 and MSA2, but the performance is the best. There was no significant difference among HSA9Fs, GMPSO and GALCS, but better than MACO. For Large-Scale, there is no significant difference among HSA9Fs, GMPSO and MSSA2, but better than GALCS. MSIA and MSSA1, however, outperform other algorithms significantly at both Medium-Small-Scale and Large-Scale, owing to our proposed resource selection method. The performance of MSSA2 dropped obviously on Large-Scale, indicating that simple rules do not adapt to the complex multi-objective optimization environments.

To analyze why MSIA performs second-best on the '5_3150' and '25_10878' test problems, we plot the parallel coordinates of the non-dominated solutions obtained by the various algorithms for these two problems, as shown in Fig. 8. According to Fig. 8(a), MSIA is superior to other algorithms in Total Tardiness and Total Energy Consumption, and inferior to GALCS in Total Monetary Cost. According to Fig. 8(b), MSIA is superior to other algorithms in Total Tardiness, and inferior to GMPSO in Total Monetary Cost and Total Energy Consumption. Compared with GMPSO, MSIA is only slightly worse in Total Monetary Cost. In general, MSIA spends more money on these two test problems



(a) $\lvert G\rvert\_\sum\lvert A^g\rvert = 5\_3150$.



(b) $\lvert G\rvert\_\sum\lvert A^g\rvert = 25\_10878$.

Fig. 8. The Parallel Coordinates Plot of non-dominated solutions obtained from different algorithms for tri-objective.

in order to complete the workflow as soon as possible. Monetary cost includes execution cost and communication cost. As HSA9Fs mainly depends on the time factor during VM selection, ignoring that data transmission may lead to cost increase.

5.3.2. Comparison of coverage rate

To quantitatively compare the dominance relationships between MSIA and its peers, we compute Coverage Rate for each test problem, as shown in Table 8. This table clearly shows that MSIA and HSA9Fs are superior to other algorithms. Especially, MSIA's solution can dominate more than 41% of the solutions of all other algorithms in Medium-Small-Scale and GMPSO in Large-Scale, but these algorithms rarely dominate MSIA. Compared with MACO, the Pareto fronts of MSIA and HSA9Fs can dominate the solution of MACO. Compared with GMPSO, GMPSO cannot dominate any solution of MSIA and HSA9Fs on all problems except on '5_1230' and '25_10878' where GMPSO can dominate a few feasible solutions of MSIA and HSA9Fs, while the Pareto front of both MSIA and HSA9Fs can dominate some solutions of GMPSO by at least 39% on average. Compared with GALCS, the Pareto front of MSIA can dominate the solution of GALCS by more than 62% on the problems of '3_200', '5_330', '5_430', '5_1300' and '15_2656', while they can barely dominate each other on other problems. There are several problems in which the solutions of MSSA1 and MSSA2 dominate MSIA' better than the solutions of MSIA dominate them, but overall MSIA's solution still dominates most of the solutions of MSSA1 and MSSA2.

Table 8
The mean of Coverage Rate of different algorithms[a].

	$\lvert G\rvert\_\sum\lvert A^g\rvert$	MSIA						HSA9Fs				
		MACO	GMPSO	GALCS	MSSA1	MSSA2	HSA9Fs	MACO	GMPSO	GALCS	MSSA1	MSSA2
Medium-Small-Scale	3_180	**0.54**/0.00	**0.63**/0.00	**0.11**/0.00	**0.47**/0.25	0.00/**0.48**	**0.25**/0.00	**0.36**/0.00	**0.35**/0.00	**0.07**/0.00	0.08/**0.25**	0.00/**0.10**
	3_200	**0.84**/0.00	**0.96**/0.00	**0.96**/0.00	**0.58**/0.16	**0.90**/0.10	**0.50**/0.00	0.00/0.00	**0.00**/0.00	**0.00**/0.00	0.00/**0.50**	0.00/**0.30**
	5_330	**0.76**/0.00	**0.96**/0.00	**0.81**/0.00	**0.74**/0.08	**0.27**/0.08	**0.40**/0.00	**0.44**/0.00	**0.59**/0.00	**0.09**/0.00	0.07/**0.15**	0.02/**0.10**
	5_430	**0.66**/0.00	**0.98**/0.00	**0.91**/0.00	**0.55**/0.14	**0.94**/0.00	**0.25**/0.00	**0.66**/0.00	**0.98**/0.00	**0.50**/0.00	0.02/**0.10**	**0.60**/0.00
	7_391	**0.81**/0.00	**0.98**/0.00	0.00/0.00	**0.74**/0.07	**0.56**/0.06	**0.50**/0.00	**0.71**/0.00	**0.15**/0.00	0.00/0.00	0.00/**0.50**	0.00/**0.10**
	7_394	**0.77**/0.00	**0.90**/0.00	**0.01**/0.00	**0.56**/0.15	**0.93**/0.00	**0.45**/0.00	**0.77**/0.00	**0.31**/0.00	0.00/0.00	0.00/**0.40**	**0.39**/0.00
	10_649	**0.50**/0.00	**0.93**/0.00	**0.37**/0.00	**0.44**/0.15	**0.24**/0.14	**0.50**/0.00	**0.10**/0.00	**0.09**/0.00	**0.01**/0.00	0.00/**0.50**	0.00/**0.50**
	10_672	**0.50**/0.00	**0.97**/0.00	**0.10**/0.00	**0.43**/0.18	**0.10**/0.05	**0.50**/0.00	**0.50**/0.00	**0.64**/0.00	0.00/0.00	0.00/**0.50**	0.00/**0.10**
	Avg	**0.67**/0.00	**0.92**/0.00	**0.41**/0.00	**0.56**/0.15	**0.49**/0.11	**0.42**/0.00	**0.44**/0.00	**0.39**/0.00	**0.08**/0.00	0.02/**0.36**	0.13/**0.15**
Large-Scale	5_1230	\	**0.71**/0.02	0.00/0.00	0.19/**0.46**	0.04/**0.29**	**0.45**/0.00	\	**0.20**/0.05	0.00/0.00	0.00/**0.50**	0.00/**0.50**
	5_1300	\	**0.98**/0.00	**0.62**/0.00	**0.33**/0.18	0.10/**0.35**	**0.05**/0.00	\	**0.97**/0.00	**0.16**/0.00	**0.01**/0.00	0.00/0.00
	5_3150	\	**0.71**/0.00	**0.00**/0.00	**0.43**/0.14	**0.29**/0.01	**0.50**/0.00	\	**0.34**/0.00	0.00/0.00	0.00/**0.45**	0.01/**0.25**
	15_2656	\	**0.98**/0.00	**0.82**/0.00	0.29/**0.32**	**0.27**/0.00	**0.50**/0.00	\	**0.38**/0.00	**0.07**/0.00	0.00/**0.45**	0.00/0.00
	15_4681	\	**0.73**/0.00	0.00/0.00	**0.59**/0.09	**0.81**/0.00	0.00/0.00	\	**0.15**/0.00	0.00/0.00	**0.01**/0.00	**0.79**/0.00
	15_5585	\	**0.72**/0.00	**0.48**/0.00	**0.52**/0.10	**0.89**/0.00	**0.50**/0.00	\	**0.43**/0.00	0.00/0.00	0.00/**0.45**	**0.03**/0.00
	25_6164	\	**0.55**/0.00	0.00/0.00	**0.51**/0.08	**0.86**/0.00	**0.45**/0.00	\	**0.34**/0.00	0.00/0.00	0.00/**0.25**	**0.02**/0.00
	25_7984	\	**0.94**/0.00	0.00/0.00	**0.45**/0.16	**0.77**/0.00	**0.45**/0.00	\	**0.72**/0.00	0.00/0.00	0.00/**0.35**	**0.02**/0.00
	25_10878	\	**0.63**/0.16	**0.02**/0.00	**0.39**/0.14	**0.85**/0.00	**0.40**/0.00	\	**0.28**/0.00	0.00/0.00	0.00/**0.25**	**0.72**/0.00
	Avg	\	**0.77**/0.02	**0.22**/0.00	**0.41**/0.19	**0.54**/0.07	**0.37**/0.00	\	**0.42**/0.01	**0.03**/0.00	0.00/**0.30**	**0.18**/0.08

[a]The meaning of $value_1/value_2$ in the column 'MSIA': $value_1$ is the proportion of MSIA that dominate other algorithms; $value_2$ is the proportion of other algorithms that dominate MSIA. The column 'HSA9Fs' is similar to this. For example, '**0.54**/0.00' is CR(MSIA,MACO)/CR(MACO,MSIA).

Table 9
Comparisons of running time and numbers of leased public cloud VMs by different algorithms.

	$\lvert G\rvert\_\sum\lvert A^g\rvert$	Given time $(1.2\sum\lvert A^g\rvert)$	Average running time (in second)							Average number of leased VMs in public cloud						
			MACO	GMPSO	GALCS	MSSA1	MSSA2	HSA9Fs	MSIA	MACO	GMPSO	GALCS	MSSA1	MSSA2	HSA9Fs	MSIA
Medium-Small-Scale	3_180	212.400	217.325	212.498	221.713	105.708	**89.386**	0.406	201.730	94	81	42	3	3	4	3
	3_200	240.000	245.243	240.119	248.637	111.978	**110.786**	0.306	202.954	83	74	42	**4**	14	**4**	3
	5_330	385.200	422.293	386.774	423.247	219.955	**165.763**	0.465	370.628	145	128	76	**7**	12	6	7
	5_430	514.800	576.085	518.474	577.469	471.664	**390.386**	1.104	472.129	168	152	92	**61**	**41**	68	**61**
	7_391	468.000	485.282	471.188	511.530	206.974	**174.252**	0.448	295.547	86	83	86	21	28	18	**20**
	7_394	470.400	486.405	472.787	524.345	310.539	**259.421**	0.675	391.558	103	97	85	20	37	26	**21**
	10_649	778.800	1599.890	795.039	894.866	640.028	**565.793**	1.861	664.872	122	118	136	66	**65**	90	61
	10_672	805.200	1763.263	820.746	980.991	598.180	**489.887**	1.226	728.641	132	127	129	33	54	**35**	**35**
	Avg	484.350	724.473	489.703	547.850	333.128	**280.709**	0.811	416.007	117	108	86	27	**32**	**32**	27
Large-Scale	5_1230	1474.800	\	1577.183	2455.390	1384.260	**1384.155**	11.594	1620.522	\	457	268	341	344	**325**	339
	5_1300	1560.000	\	1642.241	2519.676	**1466.746**	1466.803	7.601	1477.816	\	424	224	**63**	107	51	66
	5_3150	3780.000	\	5865.300	3800.684	3312.600	**3310.139**	29.513	3426.182	\	1033	859	268	310	305	**275**
	15_2656	3183.600	\	3923.040	3202.211	**3078.217**	3078.223	20.573	3146.233	\	401	510	190	297	165	**175**
	15_4681	5575.200	\	11553.736	12120.123	**4727.264**	4728.001	43.023	6380.175	\	720	1681	283	691	315	**289**
	15_5585	6264.000	\	20205.420	20207.976	6542.074	**6540.633**	87.052	8520.531	\	1087	2046	464	887	518	**493**
	25_6164	7316.400	\	24625.647	27993.986	**6298.594**	6303.554	66.396	9099.503	\	587	2189	**342**	778	297	357
	25_7984	9456.000	\	55773.033	61717.007	**8067.333**	8080.489	88.754	8669.943	\	766	2827	**406**	1507	258	436
	25_10878	12979.200	\	119692.641	162276.473	**10619.323**	10623.857	160.857	11102.164	\	937	3815	**513**	1484	320	525
	Avg	5732.133	\	27206.471	32921.503	**5055.157**	5057.317	57.262	5938.119	\	713	1603	**319**	712	284	329

Except for '15_4681', MSIA and HSA9Fs do not dominate each other, and MSIA's solution can dominate HSA9Fs' on other test problems.

5.3.3. Comparisons of running time and numbers of leased public cloud VMs

Running time is the key efficiency metric to measure the algorithm. Furthermore, while the PMWS-HC does not limit the numbers of leased public cloud VMs, employing fewer VMs reduces the possibility of failure during scheduling based on better balancing the three objectives considered. Table 9 shows comparisons of running time and numbers of leased VMs in public cloud by different algorithms.

Running time. Since HSA9Fs is a heuristic algorithm, it is obvious that its running time is minimal. The running time of MSIA is higher than MSSA1 and MSSA2 is theoretically known because MSIA uses HSA9Fs to generate the initial solution and employs IGA deep search compared to MSSA1, while MSSA2 uses a simpler decoding method than MSSA1. For Medium-Small-Scale, the running time of algorithms MACO, GMPSO, GALCS and MSIA is close to the given time, while the running times of MSSA1 and MSSA2 algorithms are much less than the given time because they have reached the given number of iterations. Particularly, the running time of MACO on '10_649' and '10_672' is much longer than the given time. This demonstrates that with the increase of scale, it becomes more difficult for MACO to get feasible solutions, which also explains why MACO cannot obtain feasible solutions on Large-Scale. For Large-Scale, as the scale increases, all algorithms have varying degrees of growth in almost all test problems when compared with the given running time. The reason lies in that with the increase of scale, it is difficult for the algorithms to escape the local optimum in a larger solution space.

Numbers of leased public cloud VMs (We abbreviate it as 'VMs counts'.). Since MSIA, MSSA1 and HSA9Fs adopt the same decoding method (Algorithm 2: ResourceSelection), their VMs counts are close and almost minimal compared with other algorithms. Since MSSA2 leases public cloud resources only when the sub-deadline is not met, it uses a similar VMs counts as MSIA on Medium-Small-Scale, but uses much more VMs counts than MSIA on Large-Scale, which is very close to GMPSO. This shows that as the number of tasks increases, it becomes increasingly difficult to complete tasks on time by relying only on private cloud resources, and the resource selection methods that do not consider many scheduling factors cannot actively select the appropriate resources. The resource selection methods of MACO, GMPSO, and GALCS all have random characteristics and do not design unique evolutionary mechanisms for resource selection, so they always have a higher VMs counts.

To summarize, the experimental results show that MSIA outperforms other algorithms in terms of the diversity and convergence of solutions, as well as the running time and the numbers of leased public cloud VMs. That is, the MSIA can better simultaneously trade off the considered three objectives.

6. Conclusion

In this paper, we propose a nested algorithm MSIA for tri-objective privacy-aware multi-workflow scheduling in hybrid cloud (PMWS-HC). To solve PMWS-HC, we devise a novel Heuristic Scheduling Algorithm based on 9 Factors (HSA9Fs), which makes the scheduling compact and non-greedy by comprehensively considering various factors such as current completion times, urgency, sub-deadline, and etc. For trading off the three objectives considered, we employ MSSA to search Pareto solution globally, and IGA to search deeply in the neighborhoods of individual to improve the quality of the solution. Extensive Medium-Small-Scale and Large-Scale simulation experiments and comparisons demonstrate that HSA9Fs and MSIA outperform state-of-the-art scheduling algorithms in several evaluation metrics. In the future work, we intend to apply HSA9Fs and MSIA in real-time multi-workflow scheduling.

CRediT authorship contribution statement

Zaixing Sun: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft, Writing – review & editing, Data curation. **Hejiao Huang:** Supervision, Conceptualization, Methodology, Resources, Project administration. **Zhikai Li:** Software, Validation. **Chonglin Gu:** Supervision, Methodology, Writing – review & editing. **Ruitao Xie:** Formal analysis, Writing – review & editing. **Bin Qian:** Formal analysis, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Ali, M., Khan, S. U., & Vasilakos, A. V. (2015). Security in cloud computing: Opportunities and challenges. *Information Sciences*, *305*, 357–383. http://dx.doi.org/10.1016/j.ins.2015.01.025.

Chen, Z. G., Zhan, Z. H., Lin, Y., Gong, Y. J., Gu, T. L., Zhao, F., Yuan, H. Q., Chen, X., Li, Q., & Zhang, J. (2019). Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach. *IEEE Transactions on Cybernetics*, *49*(8), 2912–2926. http://dx.doi.org/10.1109/TCYB.2018.2832640.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. http://dx.doi.org/10.1109/4235.996017.

Deelman, E., Gannon, D., Shields, M., & Taylor, I. (2009). Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, *25*(5), 528–540. http://dx.doi.org/10.1016/j.future.2008.06.012.

Figueiredo, E. M. N., Ludermir, T. B., & Bastos-Filho, C. J. A. (2016). Many objective particle swarm optimization. *Information Sciences*, *374*, 115–134. http://dx.doi.org/10.1016/j.ins.2016.09.026.

Hafsi, H., Gharsellaoui, H., & Bouamama, S. (2022). Genetically-modified Multi-objective Particle Swarm Optimization approach for high-performance computing workflow scheduling. *Applied Soft Computing*, *122*, Article 108791. http://dx.doi.org/10.1016/j.asoc.2022.108791.

He, F., Shen, K., Guan, L., & Jiang, M. (2017). Research on energy-saving scheduling of a forging stock charging furnace based on an improved SPEA2 algorithm. *Sustainability*, *9*(11), http://dx.doi.org/10.3390/su9112154.

Hilman, M. H., Rodriguez, M. A., & Buyya, R. (2020). Multiple workflows scheduling in multi-tenant distributed systems: A taxonomy and future directions. *ACM Computing Surveys*, *53*(1), http://dx.doi.org/10.1145/3368036.

Hu, W., Li, X., & Li, X. (2020). Hybrid cloud workflow scheduling method with privacy data. *IEEE Access*, *8*, 211540–211552. http://dx.doi.org/10.1109/ACCESS.2020.3037921.

Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., & Vahi, K. (2013). Characterizing and profiling scientific workflows. *Future Generation Computer Systems*, *29*(3), 682–692. http://dx.doi.org/10.1016/j.future.2012.08.015.

Lei, J., Wu, Q., & Xu, J. (2022). Privacy and security-aware workflow scheduling in a hybrid cloud. *Future Generation Computer Systems*, *131*, 269–278. http://dx.doi.org/10.1016/j.future.2022.01.018.

Li, Z., Ge, J., Yang, H., Huang, L., Hu, H., Hu, H., & Luo, B. (2016). A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds. *Future Generation Computer Systems*, *65*, 140–152. http://dx.doi.org/10.1016/j.future.2015.12.014.

Li, H., Xu, G., Wang, D., Zhou, M., Yuan, Y., & Alabdulwahab, A. (2022). Chaotic-nondominated-sorting owl search algorithm for energy-aware multi-workflow scheduling in hybrid clouds. *IEEE Transactions on Sustainable Computing*, *7*(3), 595–608. http://dx.doi.org/10.1109/TSUSC.2022.3144357.

Li, F., Zhang, L., Liao, T. W., & Liu, Y. (2019). Multi-objective optimisation of multi-task scheduling in cloud manufacturing. *International Journal of Production Research*, *57*(12), 3847–3863. http://dx.doi.org/10.1080/00207543.2018.1538579.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software*, *114*, 163–191. http://dx.doi.org/10.1016/j.advengsoft.2017.07.002.

Mohammad Hasani Zade, B., Mansouri, N., & Javidi, M. M. (2021). SAEA: A security-aware and energy-aware task scheduling strategy by Parallel Squirrel Search Algorithm in cloud environment. *Expert Systems with Applications*, *176*(December 2019), Article 114915. http://dx.doi.org/10.1016/j.eswa.2021.114915.

Mthunzi, S. N., Benkhelifa, E., Bosakowski, T., Guegan, C. G., & Barhamgi, M. (2020). Cloud computing security taxonomy: From an atomistic to a holistic view. *Future Generation Computer Systems*, *107*, 620–644. http://dx.doi.org/10.1016/j.future.2019.11.013.

Pan, L., Liu, X., Jia, Z., Xu, J., & Li, X. (2021). A multi-objective clustering evolutionary algorithm for multi-workflow computation offloading in mobile edge computing. *IEEE Transactions on Cloud Computing*, 1–18. http://dx.doi.org/10.1109/TCC.2021.3132175.

Pasdar, A., Lee, Y. C., & Almi'ani, K. (2020). Hybrid scheduling for scientific workflows on hybrid clouds. *Computer Networks*, *181*(February), Article 107438. http://dx.doi.org/10.1016/j.comnet.2020.107438.

Qian, B., Zhou, H. B., Hu, R., & Xiang, F. H. (2011). Hybrid differential evolution optimization for no-wait flow-shop scheduling with sequence-dependent setup times and release dates. In *Intelligent computing theories and application, 6838 LNCS* (pp. 600–611). http://dx.doi.org/10.1007/978-3-642-24728-6_81.

Qin, S., Pi, D., & Shao, Z. (2022). AILS: A budget-constrained adaptive iterated local search for workflow scheduling in cloud environment. *Expert Systems with Applications*, *198*(February 2021), Article 116824. http://dx.doi.org/10.1016/j.eswa.2022.116824.

Ren, J., Zhang, D., He, S., Zhang, Y., & Li, T. (2019). A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet. *ACM Computing Surveys*, *52*(6), 1–36. http://dx.doi.org/10.1145/3362031.

Rimal, B. P., & Maier, M. (2017). Workflow scheduling in multi-tenant cloud computing environments. *IEEE Transactions on Parallel and Distributed Systems*, *28*(1), 290–304. http://dx.doi.org/10.1109/TPDS.2016.2556668.

Rodriguez, M. A., & Buyya, R. (2014). Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing*, *2*(2), 222–235. http://dx.doi.org/10.1109/tcc.2014.2314655.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *177*(3), 2033–2049. http://dx.doi.org/10.1016/j.ejor.2005.12.009.

Saeedi, S., Khorsand, R., Ghandi Bidgoli, S., & Ramezanpour, M. (2020). Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing. *Computers & Industrial Engineering*, *147*(January), Article 106649. http://dx.doi.org/10.1016/j.cie.2020.106649.

Sahni, J., & Vidyarthi, P. (2018). A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment. *IEEE Transactions on Cloud Computing*, *6*(1), 2–18. http://dx.doi.org/10.1109/TCC.2015.2451649.

Sharif, S., Watson, P., Taheri, J., Nepal, S., & Zomaya, A. Y. (2017). Privacy-aware scheduling saas in high performance computing environments. *IEEE Transactions on Parallel and Distributed Systems*, *28*(4), 1176–1188. http://dx.doi.org/10.1109/TPDS.2016.2603153.

Sun, Z. X., Hu, R., Qian, B., Liu, B., & Che, G. L. (2018). Salp swarm algorithm based on blocks on critical path for reentrant job shop scheduling problems. In *Intelligent computing theories and application, 10954 LNCS* (pp. 638–648). http://dx.doi.org/10.1007/978-3-319-95930-6_64.

Sun, Z., Zhang, B., Gu, C., Xie, R., Qian, B., & Huang, H. (2022). ET2fa: A hybrid heuristic algorithm for deadline-constrained workflow scheduling in cloud. *IEEE Transactions on Services Computing*, 1–14. http://dx.doi.org/10.1109/TSC.2022.3196620.

van der Aalst, W. M. P., & van Hee, K. M. (2004). *Cooperative information systems, Workflow management: Models, methods, and systems*. Cambridge, MA, USA: MIT Press.

Wang, B., Wang, C., Huang, W., Song, Y., & Qin, X. (2021). Security-aware task scheduling with deadline constraints on heterogeneous hybrid clouds. *Journal of Parallel and Distributed Computing*, *153*, 15–28. http://dx.doi.org/10.1016/j.jpdc.2021.03.003.

Wang, Y., & Zuo, X. (2021). An effective cloud workflow scheduling approach combining PSO and idle time slot-aware rules. *IEEE-CAA Journal Automatica Sinica*, *8*(5), 1079–1094. http://dx.doi.org/10.1109/JAS.2021.1003982.

Wen, Y., Liu, J., Dou, W., Xu, X., Cao, B., & Chen, J. (2020). Scheduling workflows with privacy protection constraints for big data applications on cloud. *Future Generation Computer Systems*, *108*, 1084–1091. http://dx.doi.org/10.1016/j.future.2018.03.028.

Wu, H., Hua, X., Li, Z., & Ren, S. (2016). Resource and instance hour minimization for deadline constrained DAG applications using computer clouds. *IEEE Transactions on Parallel and Distributed Systems*, *27*(3), 885–899. http://dx.doi.org/10.1109/TPDS.2015.2411257.

Wu, C., & Wang, L. (2018). A multi-model estimation of distribution algorithm for energy efficient scheduling under cloud computing system. *Journal of Parallel and Distributed Computing*, *117*, http://dx.doi.org/10.1016/j.jpdc.2018.02.009.

Xia, X., Qiu, H., Xu, X., & Zhang, Y. (2022). Multi-objective workflow scheduling based on genetic algorithm in cloud environment. *Information Sciences*, *606*, 38–59. http://dx.doi.org/10.1016/j.ins.2022.05.053.

Yuan, H., Bi, J., Tan, W., Zhou, M. C., Li, B. H., & Li, J. (2017). TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds. *IEEE Transactions on Cybernetics*, *47*(11), 3658–3668. http://dx.doi.org/10.1109/TCYB.2016.2574766.

Zhou, A. C., He, B., & Liu, C. (2016). Monetary cost optimizations for hosting workflow-as-a-service in iaas clouds. *IEEE Transactions on Cloud Computing*, *4*(1), 34–48. http://dx.doi.org/10.1109/TCC.2015.2404807, arXiv:1306.6410.

Zhou, J., Wang, T., Cong, P., Lu, P., Wei, T., & Chen, M. (2019). Cost and makespan-aware workflow scheduling in hybrid clouds. *Journal of Systems Architecture*, *100*(August), Article 101631. http://dx.doi.org/10.1016/j.sysarc.2019.08.004.