

Article

Multi-Objective Security-Aware Workflow Scheduling with Soft Deadlines in Hybrid Clouds

Xinrong Zhan ¹, Junwen Lu ^{2,*}, Xiaodong Wang ², Liang Su ³ and Hangqi Zhang ⁴

¹ School of Artificial Intelligence, Xiamen Institute of Technology, Xiamen 361021, China; zhanxinrong@xit.edu.cn

² School of Computer and Information Engineering, Xiamen University of Technology, Xiamen 361024, China; xdwang@xmut.edu.cn

³ Xiamen Jinlong United Automobile Industry Co., Ltd., Xiamen 361022, China; sul@mail.king-long.com.cn

⁴ Xiamen Yaxon Networks Co., Ltd., Xiamen 361000, China; zhanghangqi@yaxon.com

* Correspondence: jwlu@xmut.edu.cn

Abstract

Hybrid clouds allow service providers to combine secure private resources with elastic public resources, but most schedulers still have three practical limitations: they commonly model task security as a binary constraint, focus on independent tasks rather than dependent workflows, and treat deadlines as hard feasibility constraints rather than service-level objectives with economic penalties. This paper studies multi-objective workflow scheduling in hybrid clouds under multi-level security and soft-deadline constraints. We formulate a multi-objective scheduling problem that captures workflow precedence, heterogeneous private servers and public virtual machines, security-level matching, inter-cloud data transfer, revenue, execution costs, and deadline-violation penalties. We then propose MSWSS, a Multi-level Security-aware Workflow Scheduling with Soft Deadlines algorithm. MSWSS first ranks workflow tasks by deadline pressure, upward rank, security level, and penalty sensitivity; it then filters resources by security clearance; it next selects resources using a profit-aware multi-objective score; and finally it performs local migration and compaction to reduce execution cost and deadline loss. Across 100 independent repetitions at each tested synthetic setting, MSWSS consistently achieves the lowest mean execution cost relative to the three adapted policies, while satisfaction, profit, and deadline violation show setting-dependent trade-offs. In a separate paired default-setting comparison, HEFT obtains slightly better service metrics while MSWSS reduces mean cost by 34.73%. A targeted high-contention ablation further shows that refinement reduces mean cost by 14.12% while decreasing satisfaction and profit and increasing the violation ratio, confirming a cost-service trade-off rather than universal dominance.

Keywords: hybrid cloud; multi-level security; multi-objective optimization; soft deadline; workflow scheduling



Academic Editor: Heming Jia

Received: 11 August 2026

Revised: 28 September 2026

Accepted: 29 September 2026

Published: 1 October 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Hybrid clouds are widely used by organizations that need both elastic capacity and control over sensitive data. A private cloud provides a trusted execution domain but has limited capacity, while a public cloud provides elastic resources at higher monetary and security-management cost. The scheduling decision therefore directly affects user satisfaction, provider profit, execution cost, deadline compliance, and security compliance.

Recent work has improved hybrid-cloud scheduling from several perspectives, including security-aware task placement [1,2], privacy-aware workflow mapping [3], cost-aware cloud bursting [4], and stochastic placement under duration variation [5]. However, three gaps remain. First, many methods use a binary security model: a task is either secure or insecure. This abstraction is too coarse for enterprise workloads, where public, internal, sensitive, confidential, and secret data often coexist. Second, independent-task models cannot capture data and control dependencies in business processes and scientific workflows. Third, hard-deadline rejection ignores the economic reality that late jobs may still be useful but incur service-level penalties.

This paper addresses these gaps by studying multi-objective workflow scheduling with multi-level security and soft deadlines. The goal is not to optimize a single metric, such as the number of completed tasks, but to balance user satisfaction, service-provider profit, execution cost, and deadline violation. This positioning extends the problem studied by Wang et al. [2] from independent tasks with binary security and hard deadlines to dependent workflows with graded security and penalty-aware soft deadlines.

The main contributions are as follows.

- We formulate a hybrid-cloud workflow scheduling problem that jointly models DAG precedence, multi-level task sensitivity, resource clearance, inter-cloud data transfer, execution cost, and soft-deadline penalty.
- We propose MSWSS, a polynomial-time heuristic that combines security-feasible resource filtering, deadline-pressure ranking, profit-aware multi-objective scoring, and local migration-based cost refinement.
- We evaluate MSWSS against three adapted policies: SeSiC [2], a newly constructed PSWS policy inspired by Lei et al. [3], and Starburst [4]. These are common-model policy adaptations rather than faithful reproductions of the original systems; in particular, PSWS is neither PSLs nor PSSA.
- We analyze the sensitivity of the proposed scheduler to deadline tightness, high-security DAG-node ratio, workflow scale, and public-cloud bandwidth.

2. Related Work

This section reviews the studies most closely related to the proposed scheduler from three perspectives: hybrid-cloud scheduling, workflow and deadline scheduling, and security-aware multi-objective optimization. The review first summarizes how prior work manages the trade-off between private capacity and public-cloud elasticity, then discusses workflow scheduling under deadline constraints, and finally examines how security and multiple service objectives have been incorporated into cloud scheduling models.

2.1. Hybrid-Cloud Scheduling

Hybrid-cloud scheduling has been studied for cost reduction, deadline satisfaction, load bursting, and energy management. Wang et al. [1] studied deadline-constrained security-aware task scheduling in heterogeneous hybrid clouds. Their method separates tasks that cannot be executed in the public cloud and then assigns expensive public-cloud tasks back to the local cluster to reduce cost. Wang et al. [2] later proposed SeSiC, which pre-schedules non-sensitive tasks to the public cloud to preserve private resources for secure tasks. This public-first idea improves finished-task satisfaction but still models security as a binary condition and focuses on independent tasks. Liang et al. [6] considered VM deployment and interval-based charging for service processes, showing that VM charging granularity can strongly affect hybrid-cloud cost. Yin et al. [5] studied stochastic hybrid-cloud placement for bag-of-tasks applications under duration uncertainty. Luo et al. [4]

proposed Starburst for cost-aware cloud bursting in production clusters; it controls job waiting and public-cloud offloading to reduce cloud spending.

These studies show that hybrid clouds require joint reasoning about private capacity and public elasticity. However, they typically emphasize either public-cloud cost or hard feasibility. In contrast, the proposed formulation considers workflow dependencies, multi-level security, and soft-deadline penalties in one model.

To make the contribution boundary explicit, MSWSS is not claimed to introduce a new general-purpose optimization principle. Its contribution is a unified, implementable scheduling formulation and heuristic for the combination of (i) DAG precedence, (ii) ordinal multi-level security matching at task and resource level, and (iii) soft deadline revenue degradation in a hybrid cloud. In contrast, Guo et al. [7] focus on hybrid-cloud workflow scheduling with a makespan-oriented objective and do not model node-level clearance matching and penalty-aware revenue in the present form. The distinction from SeSiC, PSWS, and Starburst is likewise in the joint model and the corresponding resource-selection/refinement procedure, rather than in a claim of a fundamentally new metaheuristic.

2.2. Workflow and Deadline Scheduling

Workflow scheduling has a long history in heterogeneous and cloud systems. HEFT [8] remains a widely used baseline for heterogeneous DAG scheduling because it combines upward-rank task ordering with earliest-finish-time processor selection. Mao and Humphrey [9] studied auto-scaling for cloud workflows to meet deadlines with low cost. Rodriguez and Buyya [10] proposed deadline-based provisioning for scientific workflows in IaaS clouds, while Malawski et al. [11] considered cost- and deadline-constrained provisioning for workflow ensembles. Pegasus [12] provides a practical workflow management system and illustrates the need for reliable execution across heterogeneous infrastructures. Guo et al. [7] combined dynamic neighborhood grouping with multi-objective search for hybrid-cloud workflow scheduling. Compared with these studies, this paper emphasizes security-level compatibility and service-profit degradation under soft deadlines.

2.3. Security and Multi-Objective Cloud Optimization

Security remains a central barrier to public-cloud outsourcing [13]. Lei et al. [3] considered privacy- and security-aware workflow scheduling in a hybrid cloud by combining a three-level data privacy/security model with workflow mapping. In their model, L1 data and tasks are restricted to private services, L2 data may cross cloud platforms with confidentiality and integrity protection, and L3 data require integrity protection during cross-platform transfer. They formulate a hard workflow-deadline and privacy-constrained cost-minimization problem: PSLS orders tasks by upward rank, assigns task sub-deadlines, and selects a minimum-cost feasible service, while PSSA applies simulated annealing to reorder task lists around PSLS. In contrast, the present work uses five ordinal task/resource clearance levels, a soft deadline with revenue degradation, and four reported objectives including satisfaction, profit, cost, and violation. Zhao et al. [14] used deep reinforcement learning for renewable-aware hybrid-cloud management. Zhang et al. [15] proposed an interval many-objective evolutionary algorithm for cloud task scheduling under uncertainty. Hao and Cao [16] modeled task and network dynamics in edge offloading with grey numbers, showing that uncertainty representation is important when scheduling conditions evolve during execution. Profit-aware placement [17], deadline-aware edge scheduling [18], energy-efficient stochastic scheduling [19], and energy-aware data-center management [20,21] further demonstrate the need for multi-metric decisions. CloudSim [22] is widely used to evaluate resource provisioning algorithms.

Cluster schedulers such as Quincy [23], Sparrow [24], Omega [25], Firmament [26], Quasar [27], Borg [28], and Resource Central [29] provide important design principles for large-scale resource management. Recent schedulers for modern clusters and machine-learning workloads [30–33] show the value of adaptive ranking and resource-aware scoring. MSWSS follows this systems perspective but targets security-aware hybrid-cloud workflows. To make these differences explicit, Table 1 summarizes representative studies in terms of workload type, security model, deadline treatment, objective focus, and method design.

Table 1. Comparison of representative scheduling studies.

Study	Workload	Security Model	Deadline Model	Main Objective	Method Type
HEFT [8]	DAG workflow	Not considered	Makespan-oriented	Finish time	List heuristic
Mao and Humphrey [9]	Cloud workflow	Not considered	Hard deadline	Cost with deadline	Auto-scaling heuristic
Rodriguez and Buyya [10]	Scientific workflow	Not considered	Hard deadline	Provisioning cost	Deadline partitioning
Malawski et al. [11]	Workflow ensemble	Not considered	Hard deadline	Cost and deadline	Provisioning heuristic
Wang et al. [1]	Independent tasks	Binary security	Hard deadline	Public-cloud cost	Security-aware heuristic
Lei et al. [3]	Workflow	Three-level privacy/security; encryption	Hard deadline/budget	Cost, privacy, security	PSLS + PSSA
Yin et al. [5]	Bag-of-tasks	Not considered	Deadline under uncertainty	Profit	Stochastic heuristic
SeSiC [2]	Independent tasks	Binary security	Hard deadline	User satisfaction	Three-stage heuristic
Starburst [4]	Cluster jobs	Not considered	Waiting-time control	Public-cloud cost	Production scheduler
Guo et al. [7]	Hybrid-cloud workflow	Not primary focus	Makespan-oriented	Cost, energy, makespan	Multi-objective search
MSWSS	Workflow DAG	Multi-level security	Soft deadline	Satisfaction, profit, cost, violation	Security-aware multi-objective heuristic

The comparison shows that existing studies usually address only part of the target problem. Hybrid-cloud schedulers often focus on independent tasks, public-cloud cost, or hard deadline feasibility; workflow schedulers capture DAG precedence but usually ignore node-level security labels; and security-aware methods often use binary security constraints rather than multi-level clearance matching. Therefore, the gap filled by this study is a unified scheduling model that simultaneously considers DAG workflow dependencies, per-node multi-level security requirements, soft-deadline revenue degradation, provider profit, execution cost, and deadline violation in hybrid clouds.

3. System Model and Problem Formulation

3.1. Hybrid-Cloud and Workflow Model

We consider a hybrid-cloud service provider that owns a private cloud and can rent VMs from a public cloud when local capacity is insufficient. The private cloud is trusted and suitable for security-sensitive tasks, but its capacity is limited. The public cloud provides elastic resources, but it introduces VM rental cost, network transfer cost, and lower trust unless a trusted VM type is selected. This setting reflects the common operational trade-off

in hybrid clouds: private resources are secure but scarce, while public resources are elastic but paid and less trusted.

The private cloud is represented by a resource set $\mathcal{P}$, and the public cloud provides a VM type set $\mathcal{V}$. A private server core or a public VM core is abstracted as a schedulable resource $r \in \mathcal{R}$. Each resource has computing speed g_r in GHz, read bandwidth b_r^{in} and write bandwidth b_r^{out} in MB/s, monetary cost rate c_r in USD per core-hour, and security clearance $CL_r \in \{1, 2, 3, 4, 5\}$. A larger clearance value means that the resource can process more sensitive tasks. Private resources are assigned the highest clearance, while public resources have lower clearance unless they correspond to trusted VM instances.

A user application is modeled as a workflow DAG $G = (\mathcal{T}, \mathcal{E})$, where $\mathcal{T} = \{t_1, \dots, t_n\}$ is the task set and $\mathcal{E}$ is the precedence edge set. Each DAG node t_i has computation demand a_i , input data size in_i , output data size out_i , base revenue B_i , node-level security label SL_i , soft deadline D_i , and delay penalty coefficient λ_i . The value of SL_i is attached to the node itself rather than to the whole workflow, so a single DAG may contain public, internal, sensitive, confidential, and secret tasks at the same time. An edge $(i, j) \in \mathcal{E}$ means that t_j can start only after t_i finishes and its output data are available. The DAG model is more general than an independent-task model because it captures execution precedence, node-specific security requirements, and data movement among tasks.

The security scope of this study is task-level ordinal clearance matching, not a complete mandatory information-flow policy. The label SL_i is interpreted as the minimum clearance required to execute task t_i and to handle all data made available to that task. The synthetic generator assigns this effective label directly to each node; it does not generate independent edge labels or model declassification. Consequently, Equation (1) prevents execution on an under-cleared resource but does not establish Bell–LaPadula-style noninterference between differently labelled nodes.

3.2. Security, Execution, and Profit Model

The scheduling decision is denoted by x_{ir} , where $x_{ir} = 1$ means that DAG node t_i is assigned to resource r , and $x_{ir} = 0$ otherwise. The first feasibility requirement is multi-level security matching:

$$x_{ir} = 1 \Rightarrow CL_r \geq SL_i, \quad (1)$$

which requires the clearance of a selected resource to be no lower than the sensitivity of the assigned DAG node. This model extends binary security constraints by allowing tasks and resources to be matched at different clearance levels. For example, a public node can be executed on almost any resource, while a confidential or secret node must use a private or trusted resource.

The security model is an access-control abstraction rather than a complete threat-proofness claim. Private resources are assumed trusted; a public VM is trusted for a task only when its declared clearance satisfies $CL_r \geq SL_i$. Cross-cloud transfers are assumed to use authenticated encryption, data at rest are protected by the cloud's encryption and access-control mechanisms, and VM isolation is assumed to prevent cross-tenant read access. These assumptions exclude side-channel, malicious-hypervisor, and compromised-credential attacks; such threats require additional mechanisms beyond the ordinal clearance model. In particular, encryption protects transport confidentiality but does not replace an information-flow rule; the reported security result therefore means clearance-feasible placement only.

We explicitly distinguish admission from placement. Let $y_i \in \{0, 1\}$ indicate whether task t_i is admitted. The assignment variables therefore satisfy

$$\sum_{r \in \mathcal{R}} x_{ir} = y_i, \quad x_{ir} \in \{0, 1\}. \quad (2)$$

If a task is rejected ($y_i = 0$), it generates no task-level revenue and is counted in the admission-loss term of the evaluation. For a workflow that requires complete predecessor support, rejection propagates forward through $y_j \leq y_i$ for every $(i, j) \in \mathcal{E}$; alternatively, the partial-completion semantics must be declared explicitly when node-level outputs are independently valuable. This convention prevents rejection from artificially improving cost or deadline metrics.

For each resource r , tasks assigned to that resource cannot overlap. If p is the task immediately preceding i on r , we impose $ST_i \geq FT_p$ (or the corresponding sequence-dependent setup time). For a capacity-limited resource, capacity limits the number of tasks executing concurrently, rather than the total number of tasks assigned over the complete scheduling horizon. The simulator expands each VM type into core-level resources, so each simulated core has $K_r = 1$ and can be reused sequentially after a task finishes. An effective per-core busy-time rate derived from the VM price is applied; rounded instance activation intervals are not modeled. For a formal time-indexed representation, with $z_{ijr} = 1$ when i precedes j on resource r , M a sufficiently large constant, and $\mathbf{1}(\cdot)$ the indicator function, we impose

$$ST_j \geq FT_i - M(1 - z_{ijr}) - M(2 - x_{ir} - x_{jr}), \quad (3)$$

$$ST_i \geq FT_j - Mz_{ijr} - M(2 - x_{ir} - x_{jr}), \quad (4)$$

$$\sum_{i \in \mathcal{T}} x_{ir} \mathbf{1}\{ST_i \leq t < FT_i\} \leq K_r \quad (r \in \mathcal{R}, t \geq 0). \quad (5)$$

The simulator implements this constraint through the per-core availability time: a candidate task starts at $\max(\text{predecessor-ready time}, \text{resource availability})$, and the resource availability is updated to the candidate's finish time. Thus, sequential reuse is permitted and no artificial limit is imposed on the number of tasks that a core can process over time.

The simulator uses seconds as its internal time unit. Computation demand a_i is expressed in GHz-s, data volumes in_i , out_i , and q_{ki} in MB, processor speed g_r in GHz, and bandwidth in MB/s. Consequently, every term in the following execution-time expression is measured in seconds. If task t_i is placed on resource r , its local processing time contains input reading, computation, and output writing:

$$e_{ir} = \frac{a_i}{g_r} + \frac{in_i}{b_r^{in}} + \frac{out_i}{b_r^{out}}. \quad (6)$$

When two dependent tasks are placed in different cloud domains, the successor must wait for inter-cloud data transfer. Let q_{ki} denote the edge-specific volume transferred from predecessor t_k to successor t_i . The transfer delay is modeled as

$$\tau_{ki} = \begin{cases} 0, & \text{domain}(k) = \text{domain}(i), \\ \frac{q_{ki}}{b_{r(i)}^{in}}, & \text{otherwise.} \end{cases} \quad (7)$$

where $\text{domain}(i)$ denotes whether the resource assigned to t_i is in the private or public cloud. The experiments set $q_{ki} = out_k$ for every outgoing dependency. The node input in_i denotes external input only, so predecessor output is not charged again in the input-reading term of Equation (6). The simulator uses the destination input rate as an effective end-to-end cross-domain rate; source-side limits and shared-link contention are not modeled and transfer delay within a cloud domain is neglected. The private storage rates are likewise effective per-task rates rather than an aggregate contended cluster limit. These

are simplifying assumptions rather than properties of a production network. The start and finish times satisfy

$$ST_i \geq \max_{(k,i) \in \mathcal{E}} (FT_k + \tau_{ki}), \quad FT_i = ST_i + e_{ir}. \quad (8)$$

These equations ensure that all predecessors have completed and that cross-cloud data have arrived before a task starts.

Unlike hard-deadline models, this paper assumes that a late task may still generate revenue, but the service provider pays a penalty for lateness. The realized revenue of task t_i is

$$R_i = \max\left(0, B_i - \frac{\lambda_i}{3600} \cdot \max(0, FT_i - D_i)\right). \quad (9)$$

where FT_i and D_i are in seconds and λ_i is in USD per hour. The factor $1/3600$ converts lateness from seconds to hours. This soft-deadline model is useful for service-level agreements where users may tolerate moderate delay but expect compensation. The provider profit is

$$Profit = \sum_{i=1}^n R_i - C_{pub} - C_{pri} - C_{net}, \quad (10)$$

where C_{pub} is public VM rent, C_{pri} is private operating cost, and C_{net} is public-cloud data transfer cost.

For reproducibility, the cost components and evaluation measures are defined as follows. The private-cloud cost is $C_{pri} = \sum_{r \in \mathcal{P}} c_r T_r / 3600$, where T_r is the busy time in seconds of private resource r and c_r is its hourly rate. The reported simulator converts each VM's hourly instance price p_v into the effective core-hour rate $c_v = p_v / K_v$, where K_v is its core count, and applies proportional busy-time accounting. Thus,

$$C_{pub} = \sum_{v \in \mathcal{V}_{pub}} \sum_{r \in \text{cores}(v)} c_v T_r / 3600. \quad (11)$$

This abstraction does not model rounded billing quanta, idle paid time, or joint activation intervals of cores belonging to the same VM. The incremental candidate cost used in Equation (19) is the increase in $C_{pri} + C_{pub} + C_{net}$ caused by the tentative assignment, under this proportional model. The network cost is $C_{net} = \sum_{(k,i) \in \mathcal{E}: \text{domain}(k) \neq \text{domain}(i)} (q_{ki} / 1024) \pi_{ki}$ for cross-domain transfers, where q_{ki} is the transferred output volume in MB and π_{ki} is the transfer price in USD/GB. Thus, $Cost = C_{pri} + C_{pub} + C_{net}$ and $Profit = Revenue - Cost$ are reported separately to distinguish provider cost from net economic return.

For an admitted task, normalized satisfaction is

$$s_i = \max\left(0, 1 - \frac{\max(0, FT_i - D_i)}{D_i}\right), \quad (12)$$

and the reported user satisfaction is the mean over all tasks, with rejected tasks assigned $s_i = 0$. The deadline violation ratio is

$$Vio = \frac{\sum_i [y_i = 1 \wedge FT_i > D_i] + \sum_i [y_i = 0]}{n}, \quad (13)$$

so that rejected tasks cannot be removed from the denominator. When a service contract pays only for a completed workflow, the same definitions are applied at the workflow/output-stage level; the node-level version is retained here only for experiments in which individual node outputs are declared billable.

For clarity, Table 2 summarizes the principal notation used in the system model and subsequent scheduling formulation.

Table 2. Main notation.

Symbol	Meaning
$G = (\mathcal{T}, \mathcal{E})$	Workflow DAG
SL_i	Security level of task t_i
CL_r	Security clearance of resource r
D_i	Soft deadline of task t_i (s)
B_i	Base revenue of task t_i
λ_i	Delay penalty coefficient (USD/h)
FT_i	Finish time of task t_i (s)
C_{pub}	Public-cloud renting cost

3.3. Problem Definition

The goal is to find a feasible mapping from workflow tasks to hybrid-cloud resources. A feasible solution must satisfy the atomicity constraint, precedence constraint, resource-capacity constraint, and security constraint in (1). The scheduling problem maximizes satisfaction and profit while minimizing execution cost and deadline violation:

$$\max \quad (US, Profit, -Cost, -Vio) \quad (14)$$

$$\text{s.t.} \quad \sum_{r \in \mathcal{R}} x_{ir} = y_i, \quad \forall i, \quad (15)$$

$$x_{ir} \in \{0, 1\}, \quad \forall i, r, \quad (16)$$

$$\text{precedence, capacity, and security constraints.} \quad (17)$$

Here US denotes user satisfaction, $Cost = C_{pub} + C_{pri} + C_{net}$ denotes the total execution cost, and Vio denotes the deadline violation ratio. These objectives are conflicting: using more public resources may reduce lateness but increase cost; placing more tasks in the private cloud may improve security but create contention; and reducing cost aggressively may decrease satisfaction and profit. The problem generalizes heterogeneous workflow scheduling and is NP-hard because it contains DAG scheduling on heterogeneous processors as a special case [8,34]. Therefore, this paper designs a polynomial-time heuristic rather than solving the mixed-integer formulation exactly.

Equation (14) is interpreted through the scalarized preference used by MSWSS. After min-max normalization, the scheduler minimizes a weighted sum of finish time, lateness, cost, negative realized revenue, and security mismatch. The numerical weights specified in Section 5 are reproducible settings, not a claim that the objectives are intrinsically neutral; a different service policy can select different weights. Accordingly, MSWSS is described as a weighted-sum heuristic rather than as a procedure that enumerates a Pareto front.

4. The Proposed MSWSS Algorithm

MSWSS is a static workflow-scheduling heuristic: the complete DAG and task attributes are available before upward ranks are computed. It does not model online arrivals. It avoids expensive evolutionary search while keeping the core multi-objective trade-off explicit.

4.1. Task Priority

For each task, MSWSS computes

$$\begin{aligned} Priority_i = & \alpha_1 \frac{1}{D_i} + \alpha_2 \frac{rank_i}{\max_j rank_j} + \alpha_3 \frac{B_i}{\max_j B_j} \\ & + \alpha_4 \frac{SL_i}{5} + \alpha_5 \frac{\lambda_i}{\max_j \lambda_j}, \end{aligned} \quad (18)$$

where $rank_i$ is the upward rank of the task. This priority assigns larger weights to urgent, critical-path, high-revenue, high-security, and high-penalty tasks.

4.2. Security-Feasible Resource Filtering

For each ready task t_i , MSWSS first filters resources by (1). This stage prevents the scheduler from using low-clearance public resources for sensitive tasks. It also reduces the number of candidates evaluated by the multi-objective scoring function.

4.3. Multi-Objective Resource Score

For each feasible resource r , MSWSS estimates finish time, delay, incremental cost, and expected profit. The resource score is

$$\begin{aligned} \text{Score}(i, r) = & \omega_1 \widehat{FT}_{ir} + \omega_2 \widehat{L}_{ir} + \omega_3 \widehat{Cost}_{ir} \\ & - \omega_4 \widehat{R}_{ir} + \omega_5 \widehat{\Delta}_{ir}, \end{aligned} \quad (19)$$

where $L_{ir} = \max(0, FT_{ir} - D_i)$ and $\Delta_{ir} = |CL_r - SL_i|$. Similar to the priority definition, each candidate metric is normalized within the feasible candidate set $\mathcal{R}_i$:

$$\widehat{z}_{ir} = \frac{z_{ir} - \min_{q \in \mathcal{R}_i} z_{iq}}{\max_{q \in \mathcal{R}_i} z_{iq} - \min_{q \in \mathcal{R}_i} z_{iq} + \epsilon}, \quad (20)$$

where $z_{ir} \in \{FT_{ir}, L_{ir}, Cost_{ir}, R_{ir}, \Delta_{ir}\}$ and ϵ is a small constant used to avoid division by zero. The revenue term has a negative sign because a larger expected revenue should reduce the score. The task is assigned to the feasible resource with the smallest score.

4.4. Local Refinement

After the initial mapping, MSWSS performs a cost–service refinement. It examines public-cloud tasks with high penalty sensitivity and tentatively migrates them to security-feasible private resources. After every tentative move, the complete fixed assignment is rebuilt in the same dependency-feasible task order, so that resource availability and all affected successor times are recomputed. The move is accepted only if it strictly reduces the refinement objective

$$J_{ref} = \text{Cost} + \sum_i \lambda_i \max(0, FT_i - D_i) / 3600.$$

MSWSS also compacts public VM usage by preferring early-available cores during candidate sampling.

The refinement objective differs intentionally from the normalized placement score: it targets monetary cost and an hourly lateness penalty after a feasible initial schedule has been obtained. It does not guarantee simultaneous improvement in satisfaction, profit, cost, and violation ratio. The ablation results in Section 5 therefore interpret refinement as a cost–service trade-off rather than a uniformly dominant component.

The overall procedure is summarized in Algorithm 1. MSWSS first computes the priority of all tasks and obtains a topological order that respects workflow dependencies. It then repeatedly schedules ready tasks. For each ready task, resources that do not satisfy the security clearance constraint are removed. The remaining candidates are evaluated by the multi-objective score, which jointly estimates finish time, lateness, cost, revenue, and security-level mismatch. After all schedulable tasks are mapped, a local refinement phase attempts to move selected public-cloud tasks to trusted private resources when the migration can reduce cost or deadline loss.

Algorithm 1 MSWSS: Multi-level Security-aware Workflow Scheduling with Soft Deadlines

Require: Workflow $G = (\mathcal{T}, \mathcal{E})$; resource set $\mathcal{R}$; security levels SL_i and CL_r ; deadlines D_i ; revenues B_i ; penalties λ_i

Ensure: Task-to-resource mapping $\mathcal{M}$ and schedule times ST_i, FT_i

- 1: Initialize $\mathcal{M} \leftarrow \emptyset$ and set all resources idle.
- 2: **for all** $t_i \in \mathcal{T}$ **do**
- 3: Compute upward rank $rank_i$ and priority $Priority_i$.
- 4: **end for**
- 5: Generate a dependency-feasible task list $\mathcal{L}$ by descending priority.
- 6: **while** $\mathcal{L}$ contains unscheduled ready tasks **do**
- 7: Select the ready task t_i with the highest priority.
- 8: Build feasible candidate set $\mathcal{R}_i = \{r \in \mathcal{R} \mid CL_r \geq SL_i\}$.
- 9: **if** $\mathcal{R}_i = \emptyset$ **then**
- 10: Mark t_i as rejected and continue.
- 11: **end if**
- 12: **for all** $r \in \mathcal{R}_i$ **do**
- 13: Estimate data-transfer delay, ST_{ir}, FT_{ir} , lateness L_{ir} , cost $Cost_{ir}$, and revenue R_{ir} .
- 14: Compute $Score(i, r)$.
- 15: **end for**
- 16: Select $r^* = \arg \min_{r \in \mathcal{R}_i} Score(i, r)$.
- 17: Assign t_i to r^* and update $\mathcal{M}, ST_i, FT_i$, and resource availability.
- 18: **end while**
- 19: Identify public-cloud tasks with high penalty sensitivity.
- 20: **for all** selected public-cloud task t_i **do**
- 21: Search private or trusted resources satisfying $CL_r \geq SL_i$.
- 22: Tentatively migrate t_i and rebuild the complete fixed-assignment schedule.
- 23: **if** the rebuilt schedule is feasible and strictly reduces J_{ref} **then**
- 24: Accept the migration and its recomputed task times and resource states.
- 25: **end if**
- 26: **end for**
- 27: **return** $\mathcal{M}, ST_i$, and FT_i .

Algorithm 1 can be interpreted as five functional stages. Lines 1–4 initialize the empty mapping, reset resource states, and compute the priority of each DAG node. The upward rank captures the downstream critical path, while the priority also includes deadline pressure, revenue, security level, and delay penalty; therefore, urgent and security-sensitive tasks are considered earlier. Line 5 converts these priorities into a dependency-feasible list. This step is important because a high-priority task cannot be scheduled before all of its predecessors have been placed.

Lines 6–18 form the main scheduling loop. Line 7 selects the highest-priority ready task, and line 8 removes all resources whose clearance is lower than the task security level. Thus, the node-level security requirement SL_i is enforced before any cost or deadline calculation is made. Lines 9–11 handle the rare case in which no resource can satisfy the security constraint; such a task is rejected rather than being assigned to an unsafe resource.

Lines 12–15 evaluate the remaining candidates. For each feasible resource, line 13 estimates data-transfer delay, start time, finish time, lateness, incremental cost, and expected revenue under the current partial schedule. Line 14 then applies the normalized score in (19), so that finish time, lateness, cost, revenue, and security mismatch are compared on the same scale. Lines 16–17 select the resource with the smallest score and commit the assignment by updating the mapping, task times, and resource availability. Line 18 closes the main scheduling loop.

Lines 19–26 perform local refinement after the initial schedule has been generated. Line 19 identifies public-cloud tasks with high penalty sensitivity, since delaying these tasks causes larger revenue loss. Lines 20–26 search private or trusted resources that still satisfy

the same security constraint, rebuild the full fixed-assignment schedule after a tentative migration, and accept the migration only if it strictly reduces J_{ref} . The final line returns the recomputed mapping and timing information.

If a task is rejected because no security-feasible or capacity-feasible resource exists, its unscheduled successors are marked unavailable under the complete-workflow convention. The final evaluation reports both the admitted-task metrics and the admission rate so that selective rejection is visible.

5. Experimental Evaluation

Simulations are conducted in MATLAB R2024b. Table 3 summarizes the main parameters. The private cloud is modeled as a secure local cluster with 64 aggregated cores, 2.55 GHz per-core speed, and high-throughput shared storage. The public cloud provides four VM types: three commodity VM types and one trusted VM type. The trusted VM has a higher clearance level and higher price. Five security levels are used: public, internal, sensitive, confidential, and secret.

Table 3. Simulation environment and parameter settings.

Parameter	Value	Meaning
Private cores	64	Aggregated secure local cluster
Private core speed	2.55 GHz	Secure-cluster CPU capacity
Private read/write bandwidth	6.8/6.2 GB/s	GPFS-like storage
Simulator time unit	second	All start, finish, deadline, and busy times
Data/bandwidth units	MB and MB/s	GB/s values are converted by multiplying by 1024
Hourly rate conversion	divide elapsed seconds by 3600	Penalty and compute-cost accounting
Private security level	5	Highest trust domain
VM types	4	Three commodity and one trusted type
VM core speeds	2.5, 3.6, 3.1, 2.7 GHz	Public-cloud instances
VM prices	0.0432–0.156 USD/h	On-demand VM price range
Security levels	1–5	Public, internal, sensitive, confidential, secret
Node security	$SL_i \in \{1, \dots, 5\}$	Each DAG node has a label
High-sec. node ratio	0.20–0.80	Fraction of nodes with $SL_i \geq 4$
Workflow sizes	50–260 tasks	Scale sensitivity
Repeats	100	Mean values are reported

Table 4 makes the VM-level mapping used by the simulator explicit. Each hourly instance price is divided by its core count to obtain the effective core-hour rate used in proportional busy-time accounting. The public-bandwidth experiment multiplies the common effective public transfer rate by the reported scale factor. Public storage I/O and the inter-cloud link are treated as independent effective per-task rates, so the current simulator does not model contention among simultaneous transfers.

Table 4. Public VM types used in the reported simulations.

Type	Cores	Speed/Core (GHz)	Clearance	Price (USD/h)	Cost Model	Transfer Rate
VM-1	2	2.5	2	0.0940	core busy time	scenario-scaled
VM-2	1	3.6	2	0.0432	core busy time	scenario-scaled
VM-3	2	3.1	2	0.1020	core busy time	scenario-scaled
Trusted VM	2	2.7	4	0.1560	core busy time	scenario-scaled

Workflow DAGs contain 50–260 tasks in the scale experiments. Each DAG node is assigned a security label $SL_i \in \{1, \dots, 5\}$ before scheduling. In the security-sensitivity experiment, the high-security DAG-node ratio directly controls the fraction of DAG nodes with $SL_i \geq 4$; the remaining nodes are sampled from levels 1–3. Task computation, input data, and output data follow exponential-like long-tail distributions to reflect heterogeneous workloads. Deadlines are determined by the deadline factor and task upward ranks. The underlying simulator is run for 100 independent repetitions with different random seeds, and mean values across the repetitions are reported for comparison. The fixed weights in (18) and (19) are applied after normalization.

The synthetic-instance protocol is limited to the parameter ranges reported in Table 3: workflow size 50–260 tasks, five ordinal security levels, high-security-node ratio 0.20–0.80, and deadline factors 0.9–1.8. The generator uses an edge density of 0.035 with a forward look-ahead of at most 28 task indices. For each node, computation demand is $250 + \text{Exp}(900)$ GHz-s, input data are $20 + \text{Exp}(320)$ MB, and output data are $10 + \text{Exp}(180)$ MB. A fraction specified by the high-security ratio is sampled uniformly from levels 4–5; all remaining nodes are sampled uniformly from levels 1–3. Base revenue is $0.18 + 0.00055a_i + 0.00018(in_i + out_i)$ USD, and the penalty coefficient is $0.018(0.7 + 1.4U)(1 + 0.28SL_i)$ USD per hour, where $U \sim \mathcal{U}(0,1)$. The deadline is $D_i = \phi 0.42 \sum_j a_j / (64 \times 2.55) + 0.028 rank_i + 18U$, where ϕ is the scenario deadline factor. Private operating cost is 0.012 USD per core-hour; public VM types have $(cores, clearance)$ (2,2), (1,2), (2,2), (2,4), prices (0.094, 0.0432, 0.102, 0.156) USD/h, and at most 36 VMs per type; network prices are (0.153, 0.05, 0.025, 0.08) USD/GB. Public transfer bandwidths are scaled by $\{0.5, 0.75, 1, 1.25, 1.5\}$ in the bandwidth experiment. The priority and score weights are respectively (0.75, 2.35, 1.65, 2.55, 0.35) and are applied after candidate-wise normalization; the normalization constant is MATLAB $\epsilon = 2.22 \times 10^{-16}$. Each reported value is the arithmetic mean of the corresponding metric over the 100 independent repetitions. The evidence is restricted to mean performance on static synthetic workflows with 50–260 tasks.

A unit audit was performed against the simulation implementation. The simulator stores ST_i , FT_i , D_i , execution time, transfer time, and resource busy time in seconds. It evaluates monetary delay as $\lambda_i \max(0, FT_i - D_i) / 3600$ and compute cost as $c_r T_r / 3600$. Data sizes are stored in MB and bandwidths in MB/s; values reported in GB/s are multiplied by 1024 on input, and data charged at USD/GB are divided by 1024 before multiplication by the network price.

Table 5 lists the compared methods. The three comparators are adapted policy implementations, not claimed line-by-line reproductions of their original systems. SeSiC represents the public-first security-aware scheduler from [2]. PSWS is a newly constructed greedy privacy/security-feasible workflow policy inspired by Lei et al.'s placement principle; it is not an implementation of either PSLS or PSSA. PSWS does not reproduce their L1–L3 edge-data model, encryption and digest overheads, interval billing, task sub-deadline mechanism, or simulated-annealing search. Starburst follows cost-aware cloud bursting [4]. Since the original papers use different assumptions, all methods are evaluated under the same workflow, node-security, and soft-deadline model. For every method, a candidate assignment is discarded if the resource clearance is lower than the security level of the corresponding DAG node.

Table 5. Compared scheduling methods implemented in the simulator.

Method	Representative Paper	Core Policy	Objective Emphasis
MSWSS	This work	Security-critical rank, weighted-sum scoring, refinement	Satisfaction, profit, cost, deadline
SeSiC	Wang et al. [2]	Public-first security-aware scheduling	Finished task count
PSWS (adapted)	Inspired by Lei et al. [3]	Greedy privacy/security-feasible mapping	Security-feasible placement
Starburst	Luo et al. [4]	Cluster waiting and cloud bursting policy	Public-cloud cost

For comparator transparency, Table 6 states the retained policy and the common adaptation applied to each method. Every method receives the same complete DAG, resource pool, clearance filter, soft-deadline revenue function, cost accounting, and task-level evaluation measures. The average of SeSiC, PSWS, and Starburst is used only as an aggregate descriptive reference, not as a substitute for method-by-method comparison. A clearance-filtered HEFT baseline is added in the separate minimal validation reported below. A recent multi-objective workflow scheduler and an exact MILP/CP-SAT solution for small DAGs would provide further reference points.

Table 6. Scope of the adapted baseline implementations.

Method	Retained Policy	Common Adaptation	Decision/Termination Rule
HEFT	Upward-rank ordering and earliest-finish-time resource selection	Common ordinal-clearance filter, resource pool, transfer model, and metrics	Static list scheduling ends after all reachable tasks are processed
SeSiC	Public-first placement preserving private capacity for sensitive tasks	Complete-DAG readiness, clearance filter, common soft-deadline revenue and cost model	Greedy assignment ends when every ready task is assigned or declared infeasible
PSWS (adapted)	Privacy/security-feasible workflow mapping inspired by Lei et al.	Same DAG, ordinal clearance, resource pool, revenue, cost, and rejection accounting	Greedy dependency-feasible mapping; no PSLS sub-deadline or PSSA search
Starburst	Wait-versus-burst decision emphasizing public-cloud cost	DAG-ready tasks replace independent jobs; common clearance, deadline, and metric definitions	Apply the bursting decision at each ready-task event and stop after all reachable tasks are processed

The evaluation focuses on the four scheduling objectives optimized in this paper: user satisfaction, provider profit, execution cost, and deadline violation ratio. Satisfaction is computed from deadline-normalized lateness, with rejected tasks assigned zero satisfaction. Profit is total realized revenue minus private operating cost, public VM rent, and data transfer cost. The deadline violation ratio counts both admitted tasks that finish after their deadlines and rejected tasks, with all generated tasks in the denominator, as defined in Equations (12) and (13).

The four sensitivity figures report arithmetic means over 100 independent repetitions. The Supplementary Materials include per-repetition raw CSV records for all 19 settings, including random seed, admitted-task count, rejected-task count, late-admitted-task count, the four reported metrics, and runtime. To add statistical evidence, we conducted a separate minimal validation at the default setting ($n = 120$, deadline factor 1.35, high-security ratio 0.44, and bandwidth scale 1.0). MSWSS, clearance-filtered HEFT, and three ablations were evaluated on the same 100 newly generated DAGs using fixed seeds. Table 7 reports means and 95% confidence-interval half-widths. The 19 sensitivity settings comprise five deadline-factor settings, five high-security-node-ratio settings, four workflow-size settings, and five public-bandwidth settings. Paired random-sign permutation tests used 20,000 permutations, Holm correction across 16 comparisons, and paired standardized effects d_z . Runtime was measured in MATLAB R2025b on Windows 11 with an Intel Core i7-10700 CPU and 15.8 GB RAM.

Table 7. Minimal validation at the default synthetic setting (100 paired DAGs). Values are mean $\pm$ 95% confidence interval half-width.

Method	Satisfaction	Profit (USD)	Cost (USD)	Violation	Runtime (s)
MSWSS	0.441 $\pm$ 0.009	40.03 $\pm$ 0.92	0.246 $\pm$ 0.010	0.669 $\pm$ 0.008	1.034 $\pm$ 0.060
HEFT	0.444 $\pm$ 0.009	40.44 $\pm$ 0.93	0.376 $\pm$ 0.017	0.665 $\pm$ 0.008	0.291 $\pm$ 0.002
No priority	0.441 $\pm$ 0.009	39.97 $\pm$ 0.91	0.256 $\pm$ 0.010	0.669 $\pm$ 0.008	1.225 $\pm$ 0.060
No refinement	0.441 $\pm$ 0.009	40.20 $\pm$ 0.92	0.270 $\pm$ 0.013	0.669 $\pm$ 0.008	0.313 $\pm$ 0.003
No mismatch	0.439 $\pm$ 0.009	39.70 $\pm$ 0.91	0.205 $\pm$ 0.007	0.669 $\pm$ 0.008	0.690 $\pm$ 0.044

Relative to MSWSS, HEFT improves mean satisfaction by 0.74%, profit by 1.02%, and violation by 0.52%, whereas MSWSS reduces mean cost by 34.73%. All four paired differences remain significant after Holm correction ($p_{Holm} < 0.001$), with absolute $d_z = 1.37$, 0.56, 2.14, and 0.58, respectively. MSWSS's mean runtime is 1.034 s at 120 tasks, 3.55 times HEFT's 0.291 s because every tentative refinement migration rebuilds the fixed-assignment schedule. At the default setting, refinement reduces mean cost from 0.270 to 0.246 USD (9.18%) while satisfaction decreases by 0.16%, profit decreases by 0.43%, and violation is unchanged. The cost difference is significant ($p_{Holm} < 0.001$, $d_z = 1.25$), whereas its service effects are small. Removing the mismatch term further lowers monetary cost but also lowers satisfaction and profit. Thus, the components implement explicit trade-offs rather than being uniformly superior on every metric.

To test refinement where migration opportunities are more likely, we also conducted a targeted high-contention validation on 100 paired DAGs. The configuration uses 180 tasks, 8 private cores, a deadline factor of 0.80, a high-security-node ratio of 0.60, and public-VM prices 1.5 times their default values. All other generator settings and paired seeds are unchanged. Table 8 isolates the complete MSWSS and the no-refinement variant.

Table 8. Targeted refinement validation under high contention (100 paired DAGs). Values are mean $\pm$ 95% confidence interval half-width.

Method	Satisfaction	Profit (USD)	Cost (USD)	Violation	Runtime (s)
MSWSS	0.693 $\pm$ 0.010	108.15 $\pm$ 1.92	2.571 $\pm$ 0.081	0.512 $\pm$ 0.009	1.279 $\pm$ 0.028
No refinement	0.713 $\pm$ 0.011	110.75 $\pm$ 1.99	2.993 $\pm$ 0.095	0.488 $\pm$ 0.009	0.282 $\pm$ 0.003

Under high contention, refinement reduces mean cost by 14.12%, while mean satisfaction and profit decrease by 2.80% and 2.35%, respectively, and the violation ratio increases by 4.84%. The four paired differences remain significant after Holm correction ($p_{Holm} < 0.001$), with absolute $d_z = 2.44$, 1.97, 2.58, and 2.01 for satisfaction, profit, cost, and violation, respectively. Runtime increases from 0.282 to 1.279 s because of complete-schedule reconstruction. These results demonstrate that refinement is effective for reducing monetary cost under contention, but its effect is a cost–service trade-off rather than an across-the-board improvement.

Figure 1 presents the effect of deadline tightness. Across the five factors, the mean values for MSWSS range from 0.363 to 0.512 in satisfaction, from 30.509 to 49.238 USD in profit, and from 0.224 to 0.227 USD in cost; the violation ratio ranges from 0.602 to 0.745. Relative to the mean of the three adapted policies at each setting, the average changes are +1.14% satisfaction, +1.84% profit, −52.39% cost, and +1.01% violation (where a positive violation change is unfavorable).

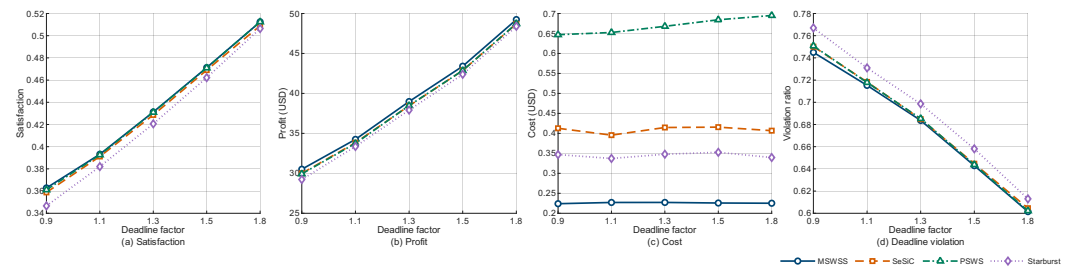


Figure 1. Performance under different deadline factors.

Figure 2 evaluates the impact of node-level security demand. The mean MSWSS values range from 0.442 to 0.452 in satisfaction, 39.819 to 41.362 USD in profit, and 0.196 to 0.262 USD in cost; violation ranges from 0.662 to 0.668. Relative to the mean baseline, the average changes are +1.21% satisfaction, +1.87% profit, −50.32% cost, and +1.18% violation (where a positive violation change is unfavorable).

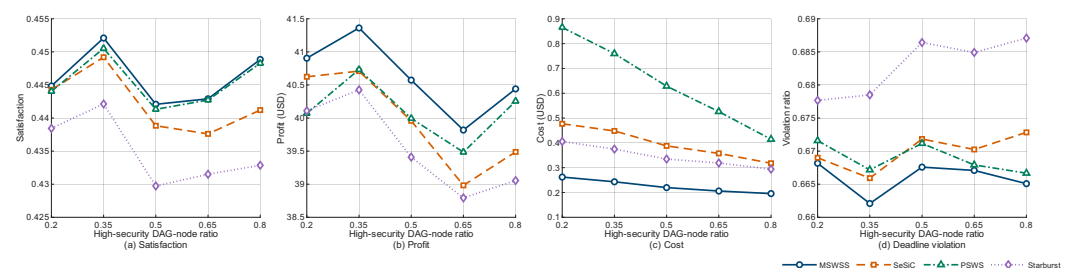


Figure 2. Performance under different high-security DAG-node ratios.

Figure 3 studies workflow scale. The mean MSWSS values range from 0.309 to 0.557 in satisfaction and from 9.243 to 123.064 USD in profit as the workflow grows from 50 to 260 tasks; cost ranges from 0.093 to 0.522 USD, while violation ranges from 0.539 to 0.804. Relative to the mean baseline, the average changes are +1.27% satisfaction, +1.95% profit, −56.71% cost, and +1.49% violation (where a positive violation change is unfavorable).

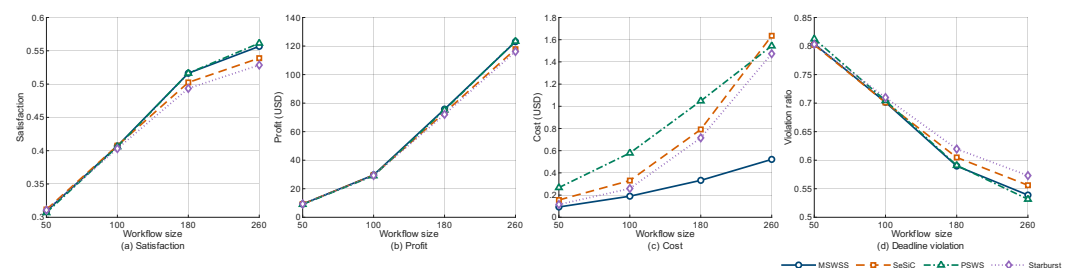


Figure 3. Performance under different workflow sizes.

Figure 4 studies public-cloud bandwidth scaling. The mean MSWSS values range from 0.444 to 0.454 in satisfaction, 40.470 to 41.211 USD in profit, and 0.191 to 0.279 USD in cost; violation ranges from 0.662 to 0.669. Relative to the mean baseline, the average changes are +1.23% satisfaction, +2.08% profit, −49.38% cost, and +1.07% violation (where a positive violation change is unfavorable).

Across the 19 tested synthetic settings, MSWSS consistently has the lowest mean execution cost and generally improves the aggregate cost–service trade-off, but it is not best on every individual service metric at every setting. The raw data show several settings in which another adapted policy has higher satisfaction or profit, or a lower violation ratio, particularly at selected workflow scales and the largest bandwidth factor. These aggregate means are descriptive and are not claims of statistical significance. The normalized score

in (19) supports a cost-aware balance by mapping finish time, lateness, cost, revenue, and security mismatch to comparable scales before resource selection.

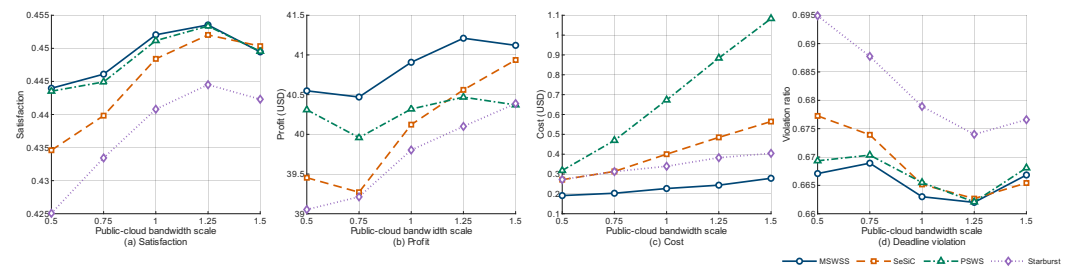


Figure 4. Performance under different public-cloud bandwidth scales.

This statement is descriptive of the present synthetic study and should not be read as a statistically proven universal dominance claim. Generalization to established workflow traces, online arrivals, and broader baseline sets remains an empirical question.

6. Conclusions

This paper presented a hybrid-cloud workflow scheduling framework that integrates multi-objective optimization, multi-level security, workflow dependencies, and soft-deadline penalties. We formulated the problem and proposed the MSWSS heuristic for security-aware soft-deadline workflow scheduling. Across 100 independent repetitions at each tested setting, the reported aggregate means show that MSWSS consistently reduces execution cost relative to the three adapted policies, while satisfaction, profit, and violation exhibit setting-dependent trade-offs. The separate paired default-setting validation with HEFT reveals a cost–service trade-off rather than dominance, whereas statistical significance across the 19 archived sensitivity settings and generalization beyond the synthetic protocol have not been established. The refinement ablation further shows a 9.18% mean cost reduction at the default setting with only small service changes, and a 14.12% cost reduction under targeted high contention accompanied by lower satisfaction and profit and a higher violation ratio. Refinement is therefore a cost-oriented local adjustment, not a mechanism that improves every objective simultaneously.

Future work will consider established workflow traces, online task arrivals, exact MILP/CP-SAT solutions for small DAGs, broader recent multi-objective baselines, weight and normalization sensitivity, fault recovery, and multiple public-cloud providers. Another useful direction is to replace fixed scoring weights with adaptive policies learned from historical SLA outcomes.

Supplementary Materials: The following supporting information can be downloaded at <https://www.mdpi.com/article/10.3390/info17100968/s1>: The default-setting validation and the high-contention refinement validation each include per-repetition data, summary statistics, and paired-test results (three CSV files per validation). The 19 sensitivity settings include per-repetition data and a summary CSV (two CSV files).

Author Contributions: Conceptualization, X.Z. and J.L.; methodology, X.Z. and J.L.; software, X.Z.; validation, X.Z. and J.L.; formal analysis, X.Z.; investigation, X.Z.; resources, X.W., L.S. and H.Z.; writing—original draft preparation, X.Z.; writing—review and editing, J.L., X.W., L.S. and H.Z.; visualization, X.Z.; supervision, J.L. and X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fujian Provincial University-Industry-Research Collaboration Innovation Project, grant number 2024H6035, and the Fujian Provincial Manufacturing Technology Innovation Key Tackling and Industrialization Project, grant number 2025G002.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The per-repetition raw and summary CSV files and paired-test outputs for the default and targeted refinement validations, together with the per-repetition raw and summary CSV files for all 19 sensitivity settings, are included in the Supplementary Materials. The sensitivity raw file contains the random seed, scenario parameters, admitted-task count, rejected-task count, late-admitted-task count, four reported metrics, and runtime for every method and repetition. Additional data are available from the corresponding author on reasonable request.

Conflicts of Interest: Liang Su and Hangqi Zhang are employed by Xiamen Jinlong United Automobile Industry Co., Ltd. and Xiamen Yaxon Networks Co., Ltd., respectively. Their employment did not influence the design, execution, interpretation, or reporting of this study. The remaining authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Wang, B.; Wang, C.; Huang, W.; Song, Y.; Qin, X. Security-aware task scheduling with deadline constraints on heterogeneous hybrid clouds. *J. Parallel Distrib. Comput.* **2021**, *153*, 15–28. [\[CrossRef\]](#)
2. Wang, B.; Cao, J.; Wang, C.; Huang, W.; Song, Y. Security-aware task scheduling for improving the user satisfaction in hybrid clouds. *J. Parallel Distrib. Comput.* **2026**, *212*, 105252. [\[CrossRef\]](#)
3. Lei, J.; Wu, Q.; Xu, J. Privacy and security-aware workflow scheduling in a hybrid cloud. *Future Gener. Comput. Syst.* **2022**, *131*, 269–278. [\[CrossRef\]](#)
4. Luo, M.; Zhuang, S.; Vengadesan, S.; Bhardwaj, R.; Chang, J.; Friedman, E.; Shenker, S.; Stoica, I. Starburst: A Cost-aware Scheduler for Hybrid Cloud. In *Proceedings of the 2024 USENIX Annual Technical Conference (USENIX ATC 24)*; USENIX Association: Berkeley, CA, USA, 2024; pp. 37–57.
5. Yin, L.; Zhou, J.; Sun, J. A stochastic algorithm for scheduling bag-of-tasks applications on hybrid clouds under task duration variations. *J. Syst. Softw.* **2022**, *184*, 111123. [\[CrossRef\]](#)
6. Liang, H.; Du, Y.; Gao, E.; Sun, J. Cost-driven scheduling of service processes in hybrid cloud with VM deployment and interval-based charging. *Future Gener. Comput. Syst.* **2020**, *107*, 351–367. [\[CrossRef\]](#)
7. Guo, Y.; Liu, B.; Lin, W.; Ye, X.; Wang, J.Z. Dynamic neighborhood grouping-based multi-objective scheduling algorithm for workflow in hybrid cloud. *Future Gener. Comput. Syst.* **2025**, *166*, 107633. [\[CrossRef\]](#)
8. Topcuoglu, H.; Hariri, S.; Wu, M.Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **2002**, *13*, 260–274. [\[CrossRef\]](#)
9. Mao, M.; Humphrey, M. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11)*, Seattle, WA, USA, 12–18 November 2011; pp. 1–12. [\[CrossRef\]](#)
10. Rodriguez, M.A.; Buyya, R. Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds. *IEEE Trans. Cloud Comput.* **2014**, *2*, 222–235. [\[CrossRef\]](#)
11. Malawski, M.; Juve, G.; Deelman, E.; Nabrzyski, J. Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. *Future Gener. Comput. Syst.* **2015**, *48*, 1–18. [\[CrossRef\]](#)
12. Deelman, E.; Vahi, K.; Juve, G.; Rynge, M.; Callaghan, S.; Maechling, P.J.; Mayani, R.; Chen, W.; da Silva, R.F.; Livny, M.; et al. Pegasus, a workflow management system for science automation. *Future Gener. Comput. Syst.* **2015**, *46*, 17–35. [\[CrossRef\]](#)
13. Subashini, S.; Kavitha, V. A survey on security issues in service delivery models of cloud computing. *J. Netw. Comput. Appl.* **2011**, *34*, 1–11. [\[CrossRef\]](#)
14. Zhao, J.; Rodriguez, M.A.; Buyya, R. A Deep Reinforcement Learning Approach to Resource Management in Hybrid Clouds Harnessing Renewable Energy and Task Scheduling. In *Proceedings of the 2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, Chicago, IL, USA, 5–10 September 2021; pp. 240–249. [\[CrossRef\]](#)
15. Zhang, Z.; Zhao, M.; Wang, H.; Cui, Z.; Zhang, W. An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty. *Inf. Sci.* **2022**, *583*, 56–72. [\[CrossRef\]](#)
16. Hao, Y.; Cao, J. Task offloading in edge computing considering the dynamics of tasks and networks based on grey numbers. *IEEE Internet Things J.* **2025**, *12*, 17742–17752. [\[CrossRef\]](#)
17. Mahmud, R.; Srirama, S.N.; Ramamohanarao, K.; Buyya, R. Profit-aware application placement for integrated Fog–Cloud computing environments. *J. Parallel Distrib. Comput.* **2020**, *135*, 177–190. [\[CrossRef\]](#)

18. Meng, J.; Tan, H.; Li, X.Y.; Han, Z.; Li, B. Online Deadline-Aware Task Dispatching and Scheduling in Edge Computing. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1270–1286. [[CrossRef](#)]
19. Li, K.; Tang, X.; Li, K. Energy-efficient stochastic task scheduling on heterogeneous computing systems. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 2867–2876. [[CrossRef](#)]
20. Barroso, L.A.; Hölzle, U. The Case for Energy-Proportional Computing. *Computer* **2007**, *40*, 33–37. [[CrossRef](#)]
21. Beloglazov, A.; Abawajy, J.; Buyya, R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Gener. Comput. Syst.* **2012**, *28*, 755–768. [[CrossRef](#)]
22. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; Rose, C.A.F.D.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [[CrossRef](#)]
23. Isard, M.; Prabhakaran, V.; Currey, J.; Wieder, U.; Talwar, K.; Goldberg, A. Quincy: Fair Scheduling for Distributed Computing Clusters. In Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP '09), Big Sky, MT, USA, 11–14 October 2009; pp. 261–276. [[CrossRef](#)]
24. Ousterhout, K.; Wendell, P.; Zaharia, M.; Stoica, I. Sparrow: Distributed, Low Latency Scheduling. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13), Farmington, PA, USA, 3–6 November 2013; pp. 69–84. [[CrossRef](#)]
25. Schwarzkopf, M.; Konwinski, A.; Abd-El-Malek, M.; Wilkes, J. Omega: Flexible, Scalable Schedulers for Large Compute Clusters. In Proceedings of the 8th ACM European Conference on Computer Systems (EuroSys '13), Prague, Czech Republic, 15–17 April 2013; pp. 351–364. [[CrossRef](#)]
26. Gog, I.; Schwarzkopf, M.; Gleave, A.; Watson, R.N.M.; Hand, S. Firmament: Fast, Centralized Cluster Scheduling at Scale. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*; USENIX Association: Berkeley, CA, USA, 2016; pp. 99–115.
27. Delimitrou, C.; Kozyrakis, C. Quasar: Resource-Efficient and QoS-Aware Cluster Management. In Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14), Salt Lake City, UT, USA, 1–5 March 2014; pp. 127–144. [[CrossRef](#)]
28. Verma, A.; Pedrosa, L.; Korupolu, M.; Oppenheimer, D.; Tune, E.; Wilkes, J. Large-scale cluster management at Google with Borg. In Proceedings of the 10th European Conference on Computer Systems (EuroSys '15), Bordeaux, France, 21–24 April 2015; pp. 1–17. [[CrossRef](#)]
29. Cortez, E.; Bonde, A.; Muzio, A.; Russinovich, M.; Fontoura, M.; Bianchini, R. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17), Shanghai, China, 28–31 October 2017; pp. 153–167. [[CrossRef](#)]
30. Narayanan, D.; Kazhamiaka, F.; Abuzaid, F.; Kraft, P.; Agrawal, A.; Kandula, S.; Boyd, S.; Zaharia, M. Solving Large-Scale Granular Resource Allocation Problems Efficiently with POP. In Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP '21), Virtually, 26–29 October 2021; pp. 521–537. [[CrossRef](#)]
31. Qiao, A.; Choe, S.K.; Subramanya, S.J.; Neiswanger, W.; Ho, Q.; Zhang, H.; Ganger, G.R.; Xing, E.P. Pollux: Co-adaptive Cluster Scheduling for Goodput-Optimized Deep Learning. In *Proceedings of the 15th USENIX Symposium on Operating Systems Design and Implementation (OSDI '21)*; USENIX Association: Berkeley, CA, USA, 2021; pp. 1–18.
32. Subramanya, S.J.; Arfeen, D.; Lin, S.; Qiao, A.; Jia, Z.; Ganger, G.R. Sia: Heterogeneity-aware, goodput-optimized ML-cluster scheduling. In Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP '23), Koblenz, Germany, 23–26 October 2023; pp. 642–657. [[CrossRef](#)]
33. Zheng, P.; Pan, R.; Khan, T.; Venkataraman, S.; Akella, A. Shockwave: Fair and Efficient Cluster Scheduling for Dynamic Adaptation in Machine Learning. In *Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation (NSDI '23)*; USENIX Association: Berkeley, CA, USA, 2023; pp. 703–723.
34. Xu, Y.; Li, K.; Hu, J.; Li, K. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. *Inf. Sci.* **2014**, *270*, 255–287. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.