

Article

DI-WOA: Symmetry-Aware Dual-Improved Whale Optimization for Monetized Cloud Compute Scheduling with Dual-Rollback Constraint Handling

Yuanzhe Kuang ^{1,2,*} , Zhen Zhang ² and Hanshen Li ³ 

¹ School of Electromechanical Engineering (New Energy), Xinjiang Agricultural University, Urumqi 830000, China

² Key Laboratory of Broadband Wireless Communication and Sensor Network, Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

³ China Electric Power Planning & Engineering Institute, Beijing 100120, China

* Correspondence: kuangyuanzhe0417@163.com

Abstract

With the continuous growth in the scale of engineering simulation and intelligent manufacturing workflows, more and more problem-solving tasks are migrating to cloud computing platforms to obtain elastic computing power. However, a core operational challenge for cloud platforms lies in the difficulty of stably obtaining high-quality scheduling solutions that are both efficient and free of symmetric redundancy, due to the coupling of multiple constraints, partial resource interchangeability, inconsistent multi-objective evaluation scales, and heterogeneous resource fluctuations. To address this, this paper proposes a Dual-Improved Whale Optimization Algorithm (DI-WOA) accompanied by a modeling framework featuring discrete–continuous divide-and-conquer modeling, a unified monetization mechanism of the objective function, and separation of soft/hard constraints; its iterative trajectory follows an augmented Lagrangian dual-rollback mechanism, while being rooted in a three-layer “discrete gene–real-valued encoding–decoder” structure. Scalability experiments show that as the number of tasks J increases, the DI-WOA ranks optimal or sub-optimal at most scale points, indicating its effectiveness in reducing unified billing costs even under intensified task coupling and resource contention. Ablation experiment results demonstrate that the complete DI-WOA achieves final objective values (OBJ) 8.33%, 5.45%, and 13.31% lower than the baseline, the variant without dual update (w/o dual), and the variant without perturbation (w/o perturb), respectively, significantly enhancing convergence performance and final solution quality on this scheduling model. In robustness experiments, the DI-WOA exhibits the lowest or second-lowest OBJ and soft constraint violation, indicating higher controllability under perturbations. In multi-workload generalization experiments, the DI-WOA achieves the optimal or sub-optimal mean OBJ across all scenarios with $H = 3/4$, leading the sub-optimal algorithm by up to 13.85%, demonstrating good adaptability to workload variations. A comprehensive analysis of the experimental results reveals that the DI-WOA holds practical significance for stably solving high-quality scheduling problems that are efficient and free of symmetric redundancy in complex and diverse environments.



Academic Editor: Shi Cheng

Received: 4 January 2026

Revised: 2 February 2026

Accepted: 3 February 2026

Published: 6 February 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](https://creativecommons.org/licenses/by/4.0/)

[Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: cloud compute scheduling; whale optimization algorithm; unified monetized objective; symmetry breaking; permutation invariance; heterogeneity-induced asymmetry; Lagrangian dual-rollback

1. Introduction

In fields such as engineering simulation, intelligent manufacturing, energy systems, and control engineering, the solution of large-scale complex models is increasingly reliant on high-performance computing platforms. Cloud computing integrates a large number of heterogeneous physical servers into elastic computing power that can be rented on demand through virtualization and resource pooling, providing a flexible computing power foundation for such engineering applications, but also bringing new problems such as rising energy consumption, uneven resource utilization, and high operation and maintenance costs [1]. This makes computing power scheduling not only about “availability” but also necessitates controllable optimization of cost and energy efficiency under service level agreement (SLA) constraints [2]. From the viewpoint of symmetry/asymmetry, cloud scheduling typically contains permutation symmetry when a subset of hosts (or VM types) are interchangeable—renumbering those resources yields alternative schedules with essentially identical physical meaning and cost—whereas real deployments are dominated by heterogeneity in performance/energy profiles and by time-varying electricity/carbon prices, which break such symmetry and require asymmetric, context-adaptive decisions. Therefore, how to stably obtain high-quality scheduling schemes under multiple constraints, heterogeneous resources, and price fluctuations has become a key capability of cloud platform operation.

The core of cloud server computing resource scheduling is to optimize metrics such as completion time, energy consumption, and economic cost by achieving the mapping and adjustment among tasks, virtual machines, and physical hosts under capacity, timing, and SLA constraints. When part of the infrastructure is homogeneous, the scheduling formulation is (approximately) invariant to permutations of those identical hosts, which induces a large set of symmetric, cost-equivalent solutions and may create plateaus in the search landscape. The problem also includes discrete decisions such as placement, migration, and power-on/off, as well as continuous variables such as frequency adjustment, and has strong coupling constraints and dynamic task arrival characteristics, which usually belong to NP-hard combinatorial optimization problems [3]. Therefore, the scheduling algorithm needs to take into account global exploration capability, feasibility improvement speed, and convergence stability, and be robust to heterogeneity-induced asymmetry such as performance diversity and price/carbon cost fluctuations [4].

Facing multi-tenant, long-time-series industrial scenarios, scheduling objectives and constraints have expanded from single performance metrics to operational costs like energy consumption, electricity/carbon price, and equipment lifespan, as well as service quality metrics like response time, violation rate, load balancing, and resource utilization [5]. However, existing research often employs fragmented rather than unified multi-objective optimization frameworks to address these diverse service quality indicators. Weighted-sum aggregation or Pareto-based ranking typically relies on manually specified preferences and inconsistent measurement scales, which breaks the “scale symmetry” (i.e., cross-scenario dimensional comparability) of evaluation and makes objective comparisons across different price systems and load regimes difficult [6]. At the modeling level, existing meta-heuristic algorithms often face an expanded search space due to the mixture of high-dimensional discrete decisions (mapping/migration/power states) and continuous DVFS variables, while ignoring the symmetry redundancy in the assignment space caused by partial resource interchangeability. This renders the feasible region sparse and the search more challenging [7].

To address these challenges, this paper proposes a Dual-Improved Whale Optimization Algorithm (DI-WOA) and its supporting modeling framework for the cloud server computing resource scheduling problem. The DI-WOA is not merely a single variant for

iterative performance enhancement of the WOA, but rather an integrated dual-rollback optimization system comprehensively equipped with the ability to perceive and handle symmetric redundancy and heterogeneous asymmetry.

To make the symmetry/asymmetry notions explicit, we summarize the main symmetry types encountered in the studied cloud scheduling model and the corresponding mitigation mechanisms in the DI-WOA. In particular, the DI-WOA targets permutation symmetry induced by interchangeable homogeneous hosts, scale symmetry caused by fragmented multi-objective aggregation with inconsistent units, and price symmetry plateaus that arise when congestion is not priced. Meanwhile, the algorithm exploits heterogeneity-induced asymmetry (e.g., non-uniform energy/performance curves and time-varying price signals) through dual-guided decoding and closed-form DVFS decisions. Various concepts related to symmetry are presented in Table 1.

Table 1. Symmetry/asymmetry types in cloud scheduling and how the DI-WOA addresses them.

Type	Where It Arises	Effect on Optimization	DI-WOA Component	What It Does
Permutation symmetry (homogeneous hosts)	Identical hosts/VM types	Many cost-equivalent schedules → redundant evaluations/plateaus	Three-layer gene-encoding-decoder + tie-breaking decoding	Maps discrete assignments into a compact continuous space and uses canonical tie-breaking to avoid exploring equivalent permutations
Scale symmetry (objective comparability)	Multi-metric aggregation w/arbitrary weights	Incomparable objectives across scenarios	Unified monetization	Converts all cost terms into “bill cost” using unit prices, yielding a dimensionally consistent single objective
Price symmetry (unpriced congestion)	Equal congestion price; early/no pricing	Decoder lacks direction in congested regions	Dual-rollback pricing	Assigns higher dual prices to violated host-slot cells → asymmetry that guides migration away from congestion
Heterogeneity-induced asymmetry (beneficial)	Hosts differ in energy/performance; time-varying price	Requires context-adaptive decisions	Dual-guided decoding + closed-form frequency	Exploits heterogeneity via slot/host-specific prices and physically consistent DVFS boundary solutions

The main contributions of the DI-WOA are as follows:

- Within the objective-function-based modeling framework, a unified monetization mechanism is adopted to convert heterogeneous cost components—such as energy consumption, electricity/carbon cost, task delay, and migration/boot-shutdown overheads—into the same “bill cost” dimension, yielding a scale-consistent (dimensionally comparable) single-objective evaluation across scenarios.
- Within the decision-variable-based modeling framework, a discrete-continuous divide-and-conquer modeling is employed; the population search is conducted only in a real-valued vector space mapped from a compact set of discrete genes (task priority, preferred host, and host-time-slot core activation ratio), while continuous frequency variables are solved analytically by a closed-form frequency step.
- Within the constraint-handling and iterative fitness evaluation modeling framework, by combining hard/soft constraint separation and an enhanced Lagrangian dual-rollback mechanism, adaptive congestion pricing is introduced to gradually assign asymmetric prices to hosts with varying congestion levels. This breaks the originally redundant, cost-equivalent allocation schemes arising from price symmetry, forming an empirical convergence trajectory characterized by “gradually increasing feasibility”.

ity rate and monotonically decreasing objective value,” thereby further improving search efficiency.

- In the encoding and decoding mechanism, the three-layer structure of “discrete genes–real-value encoding–decoder” establishes a mapping channel connecting the high-dimensional discrete search space to a low-dimensional subspace imbued with physical meaning. This cleverly transfers the DI-WOA’s search behavior to a subspace that is lower-dimensional, more continuous, and richer in physical semantics compared to the high-dimensional discrete search space. As the dimensionality of the search space is reduced, the number of symmetric redundant solutions decreases, thereby achieving implicit symmetry-based dimensionality reduction for redundant equivalent scheduling solutions.

Subsequent sections will systematically present the formal definition, implementation details, and experimental results on scalability, robustness, and generalizability of this scheduling model and the DI-WOA, with comparative validation against typical meta-heuristic algorithms.

2. Related Work

Existing research on cloud server scheduling can be categorized into three types: rule/heuristic-based methods, population intelligence and evolutionary computation-based meta-heuristic methods, and multi-strategy integrated hybrid and hyper-heuristic methods. Rule-based methods (e.g., Shortest Job First, Minimum Completion Time, and Minimum Load) are simple to implement with low computational overhead, but struggle to simultaneously balance energy consumption, cost, and SLA under strong constraints and large-scale scenarios [8]. Meta-heuristic methods (PSO, GA, ABC, Firefly Algorithm, etc.) improve solution quality and scalability through global search [9]; further, multi-objective meta-heuristic frameworks incorporate metrics like energy consumption, resource utilization, and SLA violation for joint optimization, enriching the cloud scheduling algorithm system [10]. From a symmetry/asymmetry perspective, many scheduling and placement formulations exhibit permutation symmetry when multiple machines/hosts are identical or interchangeable: renumbering such resources produces alternative schedules that are equivalent in feasibility and objective value. This symmetry enlarges the effective search space and may lead optimization methods to spend evaluations on redundant solutions. In the broader scheduling and combinatorial optimization literature, symmetry breaking has been widely studied to prune symmetric alternatives and accelerate solving, while recent discussions in optimization highlight that recognizing and leveraging symmetry can reduce redundancy and improve convergence behavior. In cloud environments, this issue coexists with heterogeneity-induced asymmetry (e.g., nonuniform performance/energy curves and dynamic price/carbon signals), which makes symmetry-aware yet asymmetry-adaptive algorithm design particularly relevant to practical schedulers.

In terms of virtual machine placement and resource configuration, a large number of studies have regarded energy consumption as one of the core optimization objectives of the scheduling problem. Some studies model VM scheduling as an energy minimization problem, aiming to reduce overall data center power consumption under performance constraints by controlling server power-on/off states, consolidating low-load VMs, and reducing migration overhead [11]. Other research proposes energy-efficiency-aware VM placement frameworks, constructing scheduling strategies based on server energy efficiency curves and utilization intervals, prioritizing high-efficiency operating regions during scheduling to lower energy consumption without significantly worsening task completion time [12]. Furthermore, multi-objective VM placement methods consider SLA violation rate and load balance alongside energy metrics, employing multi-objective population

algorithms or techniques like Quantum Particle Swarm Optimization to construct joint objectives, achieving trade-offs among energy consumption, service quality, and resource utilization [13]. In addition, the heat-aware VM placement model reveals the comprehensive impact of server layout and load allocation on energy consumption and hardware reliability by introducing factors such as temperature distribution and cooling costs, and emphasizes the importance of explicitly considering thermal constraints in energy efficiency optimization [14].

For cloud workflows and batch task scheduling with dependencies, researchers generally regard them as multi-objective optimization problems, focusing on the trade-off between execution time and energy consumption or cost. Some works construct multi-objective models for workflows, take total energy consumption and makespan as the main indicators, and use multi-objective evolutionary algorithms to search for a set of non-dominated scheduling schemes to meet deadline and resource constraints [15]. Other research, from a cost–time trade-off perspective, integrates factors like task execution cost, completion time, and SLA violation into workflow scheduling models, constructing near-optimal scheduling strategies by combining heuristics with evolutionary algorithms [16]. Based on this, some methods further introduce various biological heuristic operators and local search mechanisms to improve the convergence speed and solution set diversity of multi-objective workflow scheduling, demonstrating good adaptability in large-scale, highly coupled cloud workflow scenarios [17].

The Whale Optimization Algorithm (WOA), due to its simple structure, few parameters, and ease of integration with specific problem encoding, has found wide application in cloud task scheduling and VM placement. This algorithm simulates the bubble-net feeding behavior of humpback whales, switching between shrinking encircling mechanisms and spiral updating position to balance global exploration and local exploitation [18]. Scheduling research based on this algorithm has proposed various task–VM mapping frameworks, constructing multi-objective models primarily focusing on completion time and execution cost to obtain superior task scheduling schemes in large-scale cloud environments [19]. Subsequent work has proposed various enhancement strategies around algorithm initialization, search operators, and control parameter design, such as improving position update methods and introducing adaptive weights in edge computing scenarios to enhance convergence speed and the ability to escape local optima under complex constraints [20]. Additionally, researchers have constructed different improved Whale Optimization Algorithms for cloud task scheduling, integrating local search, mutation operators, and task-feature-guided mechanisms into the basic framework, validating their advantages in metrics like makespan and cost under various task scales and resource configurations [21]. Review studies have systematically summarized the application of the WOA and its variants in cloud task scheduling, load balancing, and workflow optimization, noting that while such methods possess advantages in solution quality and scalability, there is still room for improvement in aspects like objective modeling unity, complex constraint handling, and parameter adaptation [22].

Besides Whale Optimization Algorithms and their variants, hybrid meta-heuristic and hyper-heuristic methods for cloud task scheduling have also made significant progress in recent years. Some biogeography-based optimization algorithms incorporate migration and mutation mechanisms, combined with the TOPSIS method to calculate distances between nodes and ideal solutions, balancing execution time, energy consumption, and cost in multi-objective optimization for MEC task offloading [23]. Some cutting-edge methods combine the HS-HHO hybrid algorithm with task clustering strategies, effectively balancing latency and energy consumption for task offloading in edge-cloud collaborative scenarios, thereby improving system energy efficiency and convergence precision [24]. Some

researchers have adopted enhanced multiverse optimization algorithms in task scheduling. By introducing fitness reordering, neighborhood search, and multi-policy perturbation mechanisms in the search process, they have taken into account task completion time, cost, and resource utilization, and achieved better performance than traditional meta-heuristics on various benchmark datasets [25]. Some works have also introduced elite learning and multi-objective modeling mechanisms on the basis of the Harris Eagle Optimization Algorithm, so that the scheduling process can achieve a more balanced trade-off between multiple indicators such as load balancing, makespan, and scheduling length [26]. For green data center scenarios, some methods use multi-objective evolutionary algorithms such as fuzzy NSGA-II to closely combine DVFS technology with task scheduling models, and construct non-dominated solution sets between energy consumption, execution time, and resource utilization, so as to better meet the dual requirements of energy saving and performance [27].

Overall, existing studies have yielded rich results in cloud task scheduling and VM placement, but three common shortcomings persist:

- At the objective function level, most work employs weighted sums or Pareto ranking, relying on artificial settings for evaluation scales, making objective comparison across different price systems and load scenarios difficult;
- At the modeling level, task mapping, DVFS, and power on/off/migration decisions are often mixed within a high-dimensional discrete–continuous space, making the feasible region sparse and search difficulty high;
- At the constraint handling level, static penalties or strict hard constraints can weaken exploration capability, leading to slow feasibility rate improvement or unstable convergence.

Based on these common shortcomings of heuristic and hybrid optimization algorithms for solving cloud task scheduling problems, this paper selects the Whale Optimization Algorithm as the core skeleton of the overall optimization framework. The WOA holds the following advantages over other heuristic algorithms for cloud task scheduling problems:

- The WOA contains few hyperparameters, maintaining a lightweight structure among similar meta-heuristic algorithms, with algorithmic complexity far lower than hybrid meta-heuristics, and even simpler parameters that are easier to be compatible with various improvement mechanisms mentioned later;
- The WOA possesses a dual search mechanism, capable of balancing global exploration and local exploitation, making it easier to escape local optima compared to many early classical meta-heuristics, while achieving convergence ability comparable to complex hybrid algorithms with the simplest algorithmic structure;
- The WOA's iterative mechanism involves fewer complex operators compared to other meta-heuristics and hybrid heuristics, placing lower demands on computational resources required for iterative solving;
- Numerous hybrid algorithms and improved variants of the WOA continue to emerge, all demonstrating steady progress in cloud scheduling problems, indicating stronger extensibility for future research compared to many hybrid heuristic algorithms that have already incorporated multiple algorithmic mechanisms.

Therefore, based on the common shortcomings of existing research and the strengths of the WOA, this paper constructs a single-objective model with a unified monetized bill cost at its core, employing discrete–continuous divide-and-conquer and an enhanced Lagrange dual-rollback mechanism to gradually improve the feasibility rate while maintaining explorability, thereby enhancing solution stability and cross-scenario comparability under complex constraints.

3. Methodology

3.1. Problem Modeling and Variable Definition

This paper considers the set of discrete time slots $\mathcal{T} = 1, \dots, T$, the set of physical hosts $\mathcal{H}$, and the set of tasks $\mathcal{J}$. Task $i \in \mathcal{J}$ has an arrival slot A_i , a deadline slot D_i , a total workload W_i , and a memory requirement m_i . Host $h \in \mathcal{H}$ has a core number upper limit C_h , a memory upper limit M_h , a per-unit-frequency single-core productivity coefficient μ_h , and supports DVFS with frequency $f_{h,t} \in [f_h^{\min}, f_h^{\max}]$. The unit slot length is Δ , and the energy/carbon price is ω_t . Idle power and dynamic energy consumption coefficients are denoted as P_h^{idle} and γ_h , respectively. The booting cost is ψ_h . The unit prices for task overdue and migration/boot-shutdown costs are π_i and χ_i , respectively, with the virtualization loss coefficient $\alpha \in [0, 1)$ approximately representing a linear reduction in effective core count per active VM.

Decision variables include binary indicators and continuous allocations: host power-on indicator $y_{h,t} \in \{0, 1\}$, activated core number $c_{h,t} \in \mathbb{Z}_{\geq 0}$, frequency $f_{h,t} \geq 0$; task occupation indicator $z_{i,h,t} \in \{0, 1\}$ and workload allocation $p_{i,h,t} \geq 0$. Additionally, $b_i \geq 0$ denotes the amount of arrears at the deadline, and $r_{i,t} \geq 0$ represents the intensity of migration/boot-shutdown events during the period $t - 1 \rightarrow t$ (used for billing, not as a hard constraint).

Under the single-objective framework, a unified “bill cost” is used to measure scheduling quality. Idle and dynamic energy consumption originate from the classic DVFS mechanism (power approximated as $P \propto f^3$; under fixed allocation, the optimal frequency tends to be at the productivity boundary, and dynamic energy can be formulated as a convex term proportional to f^2 “allocated workload”). Therefore, the total cost objective function can be written as:

$$\min \sum_{h \in \mathcal{H}} \sum_{t \in \mathcal{T}} \omega_t \left(\underbrace{P_h^{\text{idle}} y_{h,t} \Delta + \frac{\gamma_h}{\mu_h} f_{h,t}^2 \sum_{i \in \mathcal{J}} p_{i,h,t}}_{\text{Energy/CarbonCost}} \right) + \underbrace{\sum_{h,t} \psi_h s_{h,t}^{\text{on}}}_{\text{BootingCost}} + \underbrace{\sum_i \pi_i b_i}_{\text{SLAViolationPenalty}} + \underbrace{\sum_{i,t} \chi_i r_{i,t}}_{\text{Migration/Boot-shutdownCost}} \quad (1)$$

The variable meanings in Equation (1) are as follows: ω_t is the unit energy/carbon price at slot t ; P_h^{idle} is the idle power of host h ; Δ is the unit slot length; γ_h is the dynamic energy coefficient of host h ; μ_h is the unit-frequency single-core productivity coefficient of host h ; $f_{h,t}$ is the frequency of host h at slot t ; $y_{h,t}$ is the on/off state of host h at slot t ; $p_{i,h,t}$ is the workload allocated to task i on host h , slot t ; ψ_h is the startup cost of host h ; $s_{h,t}^{\text{on}}$ is the power-on event indicator for host h at slot t (derived from $y_{h,t} - y_{h,t-1}$); π_i is the tardiness unit price for task i ; b_i is the unfinished workload of task i after its deadline slot; χ_i is the unit migration/on-off cost for task i ; $r_{i,t}$ is the intensity of migration/on-off events occurring at slot t . It must be emphasized that ω_t , ψ_h , π_i , and χ_i stem from price and contract parameters rather than arbitrary tunable weights, rendering the entire optimization genuinely single-objective and avoiding the subjectivity and instability of multi-objective weighting.

Figure 1 depicts the overall methodological flow of data and parameters entering the “gene $\rightarrow$ decode $\rightarrow$ closed-form frequency $\rightarrow$ cost and dual update” cyclic chain. Population search operates on the real-valued vectors mapped from discrete decisions, while continuous variables are directly provided by the decoder in a physically interpretable closed form.

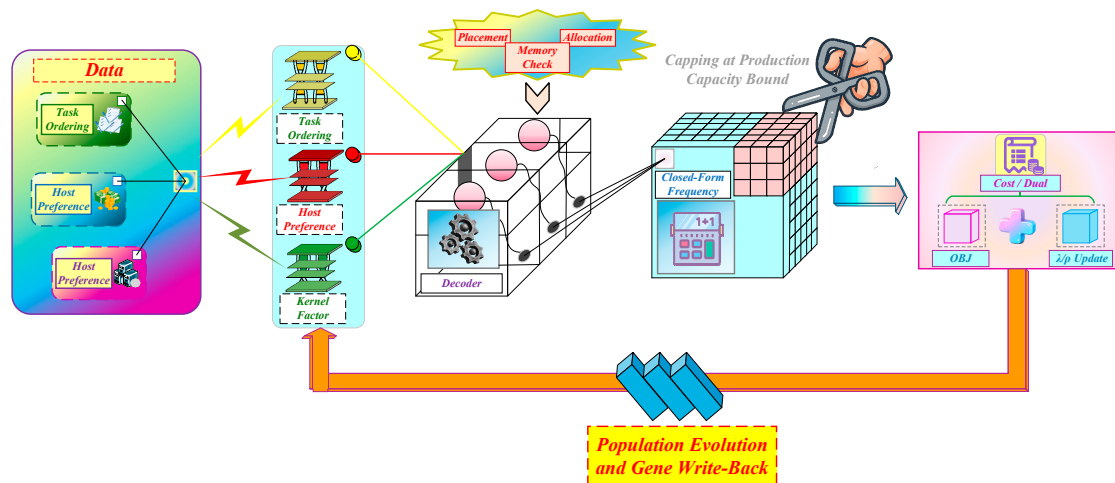


Figure 1. Cyclic chain overall methodology flowchart.

3.2. Representation Method and Feasible Decoding

Considering the sensitivity of population-based heuristics to hard constraints, to make it easier for meta-heuristic algorithms to search for feasible solutions, the constraint setup in modeling requires a soft–hard separation mechanism. This paper retains only the minimum necessary hard constraints, which must be strictly satisfied during the algorithm’s iterative solution search, including:

- A task does not run replicated in the same time slot, i.e., $\sum_h z_{i,h,t} \leq 1$
- Variable domain legality and DVFS range clipping, ensuring $f_{h,t} \in \{0\} \cup [f_h^{\min}, f_h^{\max}]$ and $c_{h,t} \in \{0, 1, \dots, C_h\}$

Soft constraints are allowed to be slightly violated during iterative solving but are subject to penalty processing via the enhanced Lagrangian mechanism after the solving process. Only one key capacity constraint is retained as soft, and the virtualization loss is endogenously folded into the effective core count, significantly reducing the number of constraints. Specifically, the concurrent VM count is defined as $vm_{s_{h,t}} = \sum_i 1[z_{i,h,t} = 1]$. The effective core number is:

$$c_{h,t}^{\text{eff}} = \max(0, \min(c_{h,t}, C_h - \alpha \cdot vm_{s_{h,t}})) \quad (2)$$

Then, the “violation amount” of the capacity soft constraint is denoted as:

$$g_{h,t}(x) = \sum_i p_{i,h,t} - \mu_h f_{h,t} c_{h,t}^{\text{eff}} \Delta \quad (3)$$

When $g_{h,t}(x) \leq 0$, it indicates that the slot’s capacity is sufficient to cover the allocation; $g_{h,t}(x) > 0$ represents an overload amount that needs to be corrected via rollback in subsequent iterations. Memory capacity constraints are enforced as hard constraints during the decoding phase. The decoder, when selecting a host and time slot for a task, first attempts placement according to the order of “preferred host $\rightarrow$ candidate hosts with lower dual prices”, but each attempt must pass a memory feasibility check. If the current host has insufficient remaining memory in that time slot, it immediately falls back to the next candidate host or later time slot, disallowing the generation of memory-exceeding solutions via “temporary occupation”. In contrast, computational capacity constraints are treated softly. Their violation amounts are used to construct the enhanced Lagrangian penalty and update dual prices, thereby guiding task workload migration towards less congested slots in subsequent iterations. Migration and boot–shutdown events are accounted for via cost terms in the bill cost rather than being treated as hard constraints.

The coupling of discrete and continuous variables generates a high-dimensional search space, causing the number of symmetric redundant solutions to increase with the rising dimensionality of the search space and rendering the feasible region sparse. Therefore, this study employs a “discrete–continuous divide-and-conquer modeling” mechanism to decouple discrete and continuous variables mixed within the same high-dimensional space, strictly splitting them into two separate solution paths for processing, thereby achieving preliminary dimensionality reduction in the search space. Discrete variables are subsequently mapped to a subspace for further dimensionality reduction. The three specific types of discrete variables include:

- Task priority sequence π ;
- Per-task preferred host vector h ;
- Host–time slot core activation ratio matrix $q \in [0, 1]^{|\mathcal{H}| \times |\mathcal{T}|}$.

Continuous variables (e.g., per-slot frequency) are not iteratively updated within the DI-WOA framework. Their feasible optimal solutions are obtained by the decoder during the evaluation phase through closed-form frequency step solving and are independently clipped on each slot, ensuring physical consistency between energy and productivity, effectively preventing blind algorithm search.

The “discrete–continuous divide-and-conquer modeling” mechanism improves decision-level decoupling and reduces the search space dimensionality, transferring the “algorithmic search framework” for cloud resource scheduling into a lower-dimensional subspace. Consequently, the number of symmetric redundant solutions decreases, playing an important role in symmetry reduction.

Figure 2 illustrates the decoder’s working mechanism. The decoder processes tasks sequentially, prioritizing the highest-priority task first, assigning it to its preferred host at its arrival slot t . If memory is insufficient at that slot, it directly falls back to other hosts with lower prices. If the preferred host has insufficient memory at that slot, it falls back to candidate hosts sorted by lower congestion prices. If the current slot’s Cap_max is less than the required capacity to handle the total task load, assuming the current task is added, then the current task is first split and placed until the current slot is filled. The remaining workload is postponed to subsequent time slots for continued filling. Subsequent Lagrangian enhanced dual terms gradually increase, forming a closed-loop guidance directing the search direction towards satisfying soft constraints as much as possible.

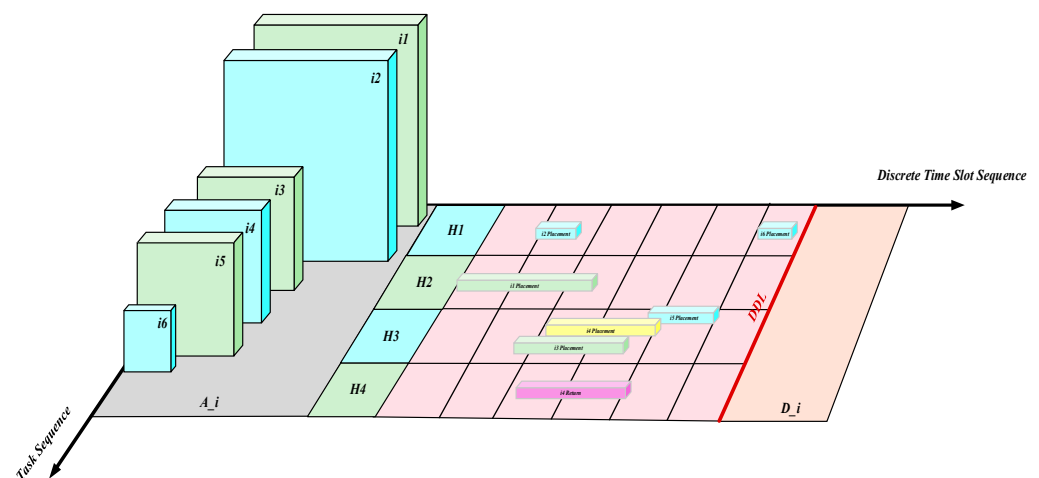


Figure 2. Working mechanism of the decoder.

The closed-form frequency step adopted in this study can “push continuous variable optimization to the boundary” given the total allocation amount $S_{h,t} = \sum_i p_{i,h,t}$ and the

effective core number $c_{h,t}^{\text{eff}}$. This avoids blind search on continuous variables, significantly reduces dimensionality, and enhances convergence stability. The optimal frequency for minimizing energy while satisfying the capacity boundary is:

$$f_{h,t}^* = \text{clip}\left(\frac{S_{h,t}}{\mu_h c_{h,t}^{\text{eff}} \Delta}, f_h^{\min}, f_h^{\max}\right) \quad (4)$$

where $\frac{S_{h,t}}{\mu_h c_{h,t}^{\text{eff}} \Delta}$ calculates the ideal optimal frequency without upper/lower bound constraints; the clip function here performs the operation of clipping the ideal optimal frequency to the permissible DVFS range $(f_h^{\min}, f_h^{\max})$. This closed-form solution ensures that, given the allocated workload and effective core count, the frequency is set to a physically feasible optimal frequency that simultaneously meets processing demands and constraints imposed by hardware performance limits, thereby eliminating the algorithm's need for iterative search over the continuous frequency variable.

3.3. Enhanced Lagrangian Dual-Rollback Mechanism

To maintain sufficient explorability of solutions during algorithm iterations, capacity soft constraints are not entirely prohibited from being violated during the decoding phase. Instead, in the evaluation phase after decoding is complete, the total penalty incurred by constraint violations in the solution space is calculated statistically as “violation amounts”. To gradually compress the violation degree while maintaining exploration, this paper incorporates capacity soft constraints into the enhanced Lagrangian framework:

$$\mathcal{L}(x, \lambda, \rho) = \text{OBJ}(x) + \sum_{h,t} \lambda_{h,t} g_{h,t}^+(x) + \frac{\rho}{2} \sum_{h,t} [g_{h,t}^+(x)]^2 \quad (5)$$

The population algorithm uses L as the fitness evaluation criterion within each generation, continuously reducing the enhanced Lagrangian value through updates to x . Between generations, λ and ρ are updated based on the violation pattern of the current best individual, thereby gradually increasing the “price” of congested slots and the cost of violating them. This paper employs a mirrored gradient ascent with a geometric scaling update rule. Let the optimal solution of generation k be $x^{(k)}$, with its corresponding violation amount $g_{h,t}^+(x^{(k)})$. The steps are:

$$\lambda_{h,t}^{(k+1)} = \max\{0, \lambda_{h,t}^{(k)} + \eta^{(k)} g_{h,t}^+(x^{(k)})\} \quad (6)$$

$$\rho^{(k+1)} = \gamma_\rho \rho^{(k)} \quad (7)$$

$$\eta^{(k+1)} = \gamma_\eta \eta^{(k)} \quad (8)$$

where $\eta^{(k)}$ is the dual step scale, and γ_ρ and γ_η control the amplification of the penalty coefficient and the decay of the step size, respectively. Intuitively, slots that are more frequently violated acquire higher dual prices. Consequently, in subsequent decoding and task filling processes, they are considered “expensive resources” due to increased penalty terms, and the decoder tends to migrate tasks towards lower-price regions. As iterations proceed, the increase in ρ drastically amplifies any residual violation in the objective, prompting the algorithm to focus its efforts in later stages on searching for local optima near feasible solutions, presenting the empirical convergence trajectory of “feasibility rate rising, objective monotonically decreasing.”

Figure 3 illustrates the complete process where discrete genes are decoded into specific scheduling solutions, followed by the update of the λ heatmap based on the capacity

constraint violation amounts of each host in the specific scheduling solution. Discrete genes are input into the decoder to generate a concrete scheduling solution (referring to the current λ heatmap). Subsequently, the capacity soft constraint violation amount for each host in the concrete scheduling solution is tallied. Finally, based on the tallied capacity soft constraint violation amounts, the λ heatmap is updated by increasing the dual prices for hosts with larger violation amounts.

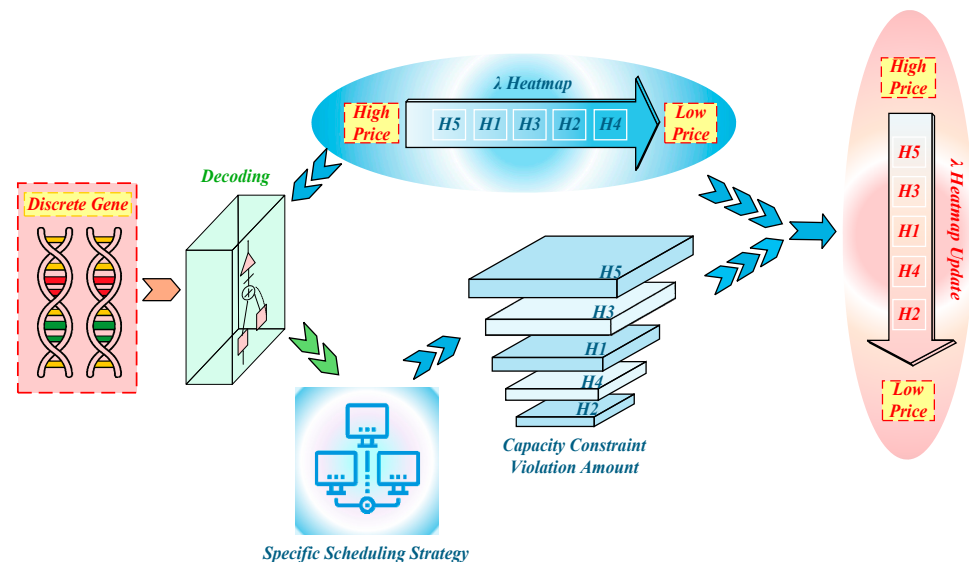


Figure 3. Update mechanism for penalty unit prices of soft constraint violations.

Overall, the enhanced Lagrangian dual-rollback mechanism is essentially a progressive constraint-handling mechanism of “exploration–guidance–convergence.” In early iterations, penalty charges for soft constraint violations on each host are low, allowing a small number of tasks to exceed deadlines. The iterative trajectory explores mildly infeasible regions, maintaining algorithm friendliness towards early exploration and avoiding premature convergence to local optima due to overly restrictive constraints on the search space. As the number of iterations increases, the dynamic dual pricing mechanism gradually raises penalty charges for soft constraint violations on congested hosts, avoiding “sudden pressure” and guiding the algorithm with sufficient time to transition from “allowing trial and error” to “satisfying constraints.” The feasibility rate gradually improves, strengthening stability in constraint handling. In later iterations, penalty charges for soft constraint violations on severely congested hosts have risen to extremely high values. Violating capacity soft constraints incurs severe penalties, forcing the iterative trajectory to stably converge towards feasible solutions. At this stage, the constraints exert sufficient restrictiveness and stability.

3.4. Dual-Improved Whale Optimization Algorithm

To reduce search variable dimensionality, diminish symmetric redundant solutions, and alleviate feasible region sparsity, this paper selects the technical chain of “discrete decision abstraction → low-dimensional real-value mapping → physical feasible solution restoration” to achieve DI-WOA search space dimensionality reduction. Based on this technical chain, a three-layer structure of “discrete gene–real-value encoding–decoder” is constructed. The specific roles of each layer are as follows:

- The discrete gene layer abstracts the core decision elements of cloud scheduling into physical representations, specifically comprising: task priority ordering order, task preferred host vector $host_pref$, and host–slot core activation ratio matrix $core_ratio$.

This layer retains only the key discrete variables affecting scheduling solution quality, enabling preliminary decoupling and dimensionality reduction in the variable space in conjunction with the discrete–continuous separation mechanism, as well as preliminary implicit symmetry reduction for redundant solutions.

- The core function of the real-value encoding layer is to map discrete genes into continuous real-value vectors via a random key mechanism. On the one hand, a random key vector of length $|J|$ is used to sort tasks; on the other hand, the preferred host index is normalized to the $[0,1]$ interval, and the core ratio matrix is flattened in row-major order, ultimately yielding a real-valued vector $x \in [0,1]^d$ of length $d = 2|J| + |H||T|$. This layer further reduces the discrete variable space to a lower-dimensional subspace, facilitating algorithm search within a lower-dimensional continuous search space and achieving further implicit symmetry reduction for redundant solutions.
- The decoder layer is the key component for transforming abstract search results into concrete, feasible scheduling solutions. Specifically, it first inversely maps the encoding vector back to discrete decisions via a decoding function; then, combined with closed-form frequency calculation and constraint checking, it generates high-quality feasible scheduling solutions that satisfy hard constraints and are adapted to soft constraints.

Overall, this three-layer structure establishes a mapping channel connecting the high-dimensional discrete search space with a physically meaningful low-dimensional subspace, transferring algorithm search into a low-dimensional and physically meaningful subspace. It reduces redundant symmetric representations caused by interchangeable resources, achieving implicit symmetry reduction for redundant equivalent scheduling solutions. Simultaneously, in conjunction with the discrete–continuous separation mechanism, it achieves preliminary decoupling and dimensionality reduction in the variable space. This holds strong practical value for enhancing the DI-WOA's convergence stability and ability to search for high-quality feasible solutions.

The DI-WOA search is executed in the real-valued vector space through iterative updates of the Whale Optimization Algorithm. Each fitness evaluation undergoes the complete chain: “real-valued vector $\rightarrow$ discrete genes $\rightarrow$ scheduling decoding $\rightarrow$ closed-form frequency and enhanced Lagrangian calculation”. Specifically, an initial instance is first constructed using a random problem generator. Based on this, a heuristic initial gene is generated and encoded via an encoding function to obtain a relatively good starting point. The remaining individuals are then initialized uniformly at random, forming an initial population possessing both diversity and heuristic characteristics. Subsequently, the main loop begins. In generation k , for each individual $x_i^{(k)}$, truncation is applied to ensure it lies within $[0,1]^d$. Then, the decoding function recovers the discrete genes and the complete schedule, calculating OBJ, various cost items, violation degrees, and the enhanced Lagrangian value, thereby selecting the best individual of that generation and its corresponding violation pattern.

In the position update phase, the DI-WOA employs the I-WOA's two modes, “shrinking encircling prey” and “spiral updating”, and introduces a low-probability Gaussian perturbation on this basis to enhance diversity. Let the global best position vector in generation k be X^* , and the current individual position be $X_i^{(k)}$. With the maximum iteration number K , the linearly decreasing coefficient is:

$$a^{(k)} = 2 \left(1 - \frac{k}{K-1} \right) \quad (9)$$

Then, generate random numbers r_1, r_2, p, l and construct:

$$A = 2a^{(k)}r_1 - a^{(k)} \quad (10)$$

$$C = 2r_2 \quad (11)$$

When $p < 0.5$ and $|A| < 1$, execute shrinking encircling around the current global best solution:

$$X_i^{(k+1)} = X^* - A \odot |C \odot X^* - X_i^{(k)}| \quad (12)$$

When $p < 0.5$ and $|A| \geq 1$, explore around a random individual. When $p \geq 0.5$, execute spiral updating towards the best solution:

$$X_i^{(k+1)} = |X^* - X_i^{(k)}| \exp(bl) \cos(2\pi l) + X^* \quad (13)$$

where b is the spiral coefficient and A takes a value in $[-1, 1]$. To prevent premature population contraction into a narrow region, the DI-WOA also superimposes a zero-mean Gaussian perturbation with a certain probability after updating and clips the result again to the $[0, 1]^d$ interval. Through this design, a loose coupling relationship is formed between the search in the real-valued vector space and the underlying discrete scheduling structure: the WOA is responsible for balancing exploration and exploitation in the encoded space, while the decoder and the enhanced Lagrangian framework ensure the physical reasonableness and feasibility of solutions.

Figure 4 illustrates the complete process wherein three types of continuous vectors obtained after DI-WOA optimization iteration are fed into the decoder for decoding, the capacity constraint violations of each host are tallied to update dual prices, thereby guiding the search direction for the next iteration of the DI-WOA. The three continuous real-value vectors derived from the DI-WOA search are preliminarily inversely mapped into three types of discrete genes via specific rules of the decoding function. These discrete genes are then further decoded to obtain concrete scheduling solutions and the capacity soft constraint violation amounts for each host. Based on the tallied capacity soft constraint violation amounts, the dual prices for hosts with larger violations are increased. The updated dual prices for all hosts are aggregated and sorted, further guiding the update of parameters fed into the enhanced Lagrangian for the next algorithm iteration, thereby dual guiding the search direction for the next iteration of the DI-WOA.

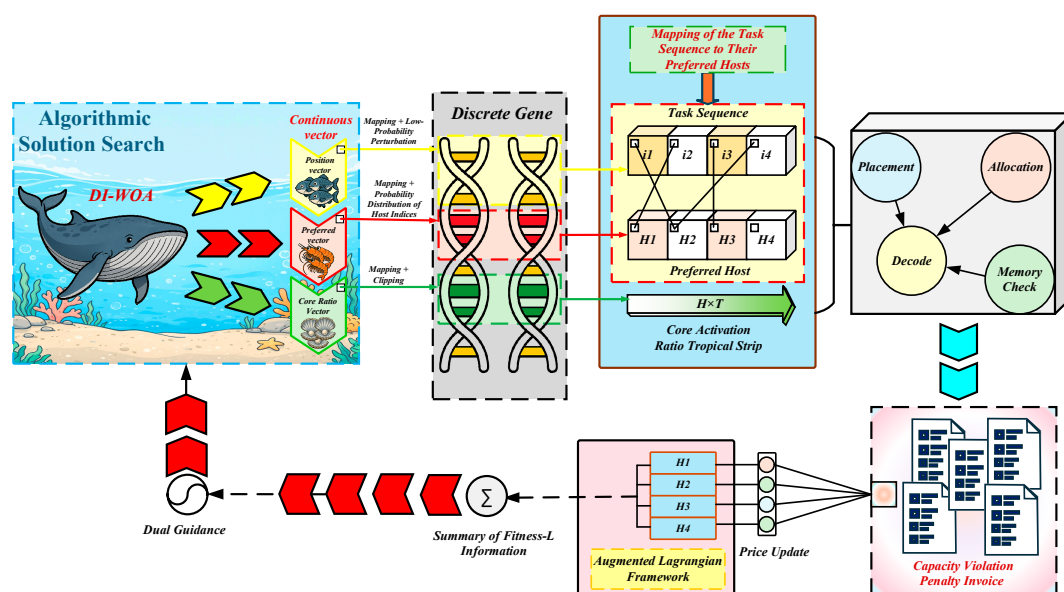


Figure 4. Overall optimization process of the DI-WOA framework.

3.5. Complexity

We analyze the computational complexity of the DI-WOA by decomposing one fitness evaluation into its major modules and then aggregating the per-iteration costs. Let J be the number of tasks, H the number of physical hosts, T the number of discrete time slots, P the population size, and G the number of generations. The DI-WOA performs the population search in a real-valued encoded space whose dimension is

$D = J$ (random keys for task ordering) + J (preferred host vector) + $H \cdot T$ (flattened core activation ratio), i.e., $D = 2J + H \cdot T$.

Per-solution (fitness) evaluation. For a candidate vector x , one evaluation consists of:

- (1) Gene recovery and task ordering. Recovering the discrete genes from x includes sorting the J random keys to obtain a task order, which costs $O(J \log J)$, plus linear-time mappings for the preferred host vector and the $H \times T$ core ratio grid, i.e., $O(J + H \cdot T)$.
- (2) Feasible decoding on the host–time grid. The decoder schedules tasks sequentially. In the worst case, each task may test multiple host–slot candidates across the $H \times T$ grid (with constant-time feasibility checks such as memory checks and capacity bookkeeping). Therefore, the decoding cost is $O(J \cdot H \cdot T)$ in the worst case.
- (3) Closed-form frequency step. Given the decoded allocation and effective core counts, the DVFS frequency is computed with a closed-form expression and clipping for each host–slot cell, which costs $O(H \cdot T)$.
- (4) Cost and violation statistics. Bill cost components (idle/dynamic energy, boot–shutdown, migration, and tardiness) are aggregated and the soft-constraint violation map is computed over the grid costs $O(H \cdot T + J)$ once the decoded schedule is available.

Combining the above, the time complexity of one fitness evaluation can be expressed as

$$C_{eval} = O(J \cdot H \cdot T + J \log J + H \cdot T) \quad (14)$$

In typical medium-scale settings, the decoding term $O(J \cdot H \cdot T)$ dominates.

Per-generation complexity. Each generation evaluates P individuals and updates their positions in the D -dimensional encoded space. The WOA position update (including the optional low-probability perturbation) is composed of element-wise vector operations and random number generation, costing $O(P \cdot D)$ per generation. In addition, the DI-WOA updates the dual variables (e.g., the congestion price heatmap λ and penalty coefficient ρ) once per generation based on the best individual's violation map, which costs $O(H \cdot T)$. Therefore, the per-generation complexity of the DI-WOA is

$$\begin{aligned} C_{iter}(DI - WOA) &= O(P \cdot C_{eval} + P \cdot D + H \cdot T) \\ &= O(P \cdot (J \cdot H \cdot T + J \log J + H \cdot T) + P \cdot D + H \cdot T) \end{aligned} \quad (15)$$

Total complexity. Over G generations, the total time complexity is

$$C_{total}(DI - WOA) = O(G \cdot C_{iter}(DI - WOA)) \quad (16)$$

We conducted a comparison with other meta-heuristics. In our experiments, all compared algorithms share the same encoding–decoding and fitness-evaluation pipeline; hence, they share the same dominant evaluation cost C_{eval} . Their overall complexities differ mainly in (i) the number of fitness evaluations per iteration and (ii) the per-iteration position-update overhead in the D -dimensional space. Most population-based methods, such as the WOA/HHO/SMA, evaluate approximately P candidates per iteration and perform $O(P \cdot D)$ updates, yielding $O(P \cdot C_{eval} + P \cdot D)$ per iteration. In contrast, ABC-type methods (including IABC) commonly perform both employed-bee and onlooker-bee phases,

which can result in roughly $2P$ fitness evaluations per cycle (implementation-dependent), leading to a higher evaluation-dominated cost. The DI-WOA introduces an extra $O(H \cdot T)$ dual-update step per generation, but this overhead is typically negligible compared with the dominant $O(P \cdot J \cdot H \cdot T)$ decoding cost.

4. Experiments

This chapter systematically evaluates the solution quality, computational efficiency, and stability of the DI-WOA under the unified bill cost objective. Experiments include four categories: scalability (task size growth), robustness (input perturbation), ablation (contribution of key mechanisms), and generalization (different load/resource configurations). Unless otherwise specified, all compared algorithms run under the same encoding–decoding and fitness evaluation framework, reporting metrics like OBJ and runtime to ensure fairness and reproducibility of comparison. Experiments were conducted on a Lenovo 20TQA004CD platform (Intel i7-10750H, 16 GB RAM). Algorithms were implemented in Python (3.10 edition) and run in a PyCharm (2025.2 edition) environment.

4.1. Experimental Setup

4.1.1. Hyperparameter Settings

To ensure fair comparison, this paper uniformly sets computational budgets (e.g., population size and iteration count) for all algorithms and runs them under the same encoding–decoding and fitness evaluation framework; algorithm-specific parameters are fixed according to common values or literature recommendations. Table 2 summarizes the main experimental parameters and problem scale settings.

Table 2. Summary of experimental parameter settings.

Hyperparameter	Value	Applicable Algorithm
pop_size	30	All experimental algorithms
generations	60	All experimental algorithms
elite_frac	0.2	All experimental algorithms
mutation_prob	0.4	All experimental algorithms
step_scale	0.15	I-HOA
rho_growth	1.2	DI-WOA, I-HOA
eta0	0.5	All experimental algorithms
eta_decay	0.9	All experimental algorithms
seed	42	All experimental algorithms
phi_scale	1.0	IABC
perturb_prob	0.1	DI-WOA
rho_penalty	1.0	HHO, SMA, IABC, RSA, HOA
rho0	1.0	DI-WOA, I-HOA

4.1.2. Compared Algorithms and Performance Evaluation Metrics

To further objectively evaluate the performance of the Dual-Improved Whale Optimization Algorithm (DI-WOA), this section selects seven meta-heuristic algorithms and hybrid meta-heuristic algorithms that have exhibited excellent performance in various engineering optimization scenarios for comparative validation against the DI-WOA. These are: the Hippopotamus Optimization Algorithm (HOA), the Improved Hippopotamus Optimization Algorithm (I-HOA), the Slime Mould Algorithm (SMA), the Improved Artificial Bee Colony Algorithm (IABC), Harris Hawks Optimization (HHO), the Reptile Search Algorithm (RSA), and the Hybrid Improved Weak Cooperation Multi-objective Dung Beetle Optimization Scheduling Algorithm (HCMDBOA).

The performance metrics evaluated in this experiment include:

- Robustness to perturbations;
- Scalability of the algorithm when facing larger-scale problems;
- Effectiveness of individual and combined innovative points of the algorithm;
- Generalization of the algorithm under various load/scale configurations when facing cross-scenario problems.

To facilitate subsequent quantification of the DI-WOA's performance metrics, relevant relative performance indicators, their physical meanings, and evaluation criteria are given here, summarized in Table 3. (In the table, the symbol “↑” indicates that a higher value of the relative performance indicator is considered better in the evaluation system.)

Table 3. Summary table of relative performance indicator meanings and evaluation criteria.

Relative Performance Indicator	Algorithm Physical Definition (%)
Bill Cost Relative Saving Rate ↑	Percentage reduction in the bill cost of the DI-WOA compared to a contrast algorithm
Penalty Cost Relative Saving Rate ↑	Percentage reduction in the penalty cost of the DI-WOA compared to a contrast algorithm
Running Time Relative Reduction Rate ↑	Percentage reduction in total computation time for completing all iterations of the DI-WOA compared to a contrast algorithm
Convergence Speed Relative Acceleration Rate ↑	Percentage reduction in iterations required for the DI-WOA to reach a stable optimal solution compared to a contrast algorithm
Bill Cost Stability Relative Increase Rate ↑	Percentage reduction in the box range of the bill cost for the DI-WOA compared to a contrast algorithm
Penalty Cost Stability Relative Increase Rate ↑	Percentage reduction in the box range of the penalty cost for the DI-WOA compared to a contrast algorithm

4.2. Experimental Design and Result Analysis

The following subsections present for each experiment type: purpose and setup, main results (figures/tables), and cause analysis oriented towards the model and algorithm mechanisms. Besides OBJ, the robustness experiment additionally focuses on fluctuations of soft constraint penalty terms; the ablation experiment explains the contribution of key mechanisms to the convergence trajectory and final solution quality.

4.2.1. Scalability Experiment and Result Analysis

The scalability experiment evaluates the trend of changes in the DI-WOA's bill cost (OBJ) and runtime as the task scale increases, to examine its scalability in large-scale scheduling scenarios. The experiment fixes host number $H = 4$, time slot number $T = 24$, and other parameters (see Table 2), taking task number J as the sole independent variable, incrementing from 10 to 100 (step size 10). For each J , each algorithm runs within the same iteration budget, recording final OBJ and runtime to compare solution quality and computational cost under different task scales.

The results of the scalability experiment are shown in Figure 5 (final OBJ) and Figure 6 (runtime). Figure 5 shows that as the number of tasks J increases, the OBJ of each algorithm generally increases; among them, the DI-WOA and HCMDBOA are in the optimal or sub-optimal sequence at most scale points, indicating that they can still effectively reduce the unified billing cost under high task coupling and enhanced resource contention. When $J = 60$, the DI-WOA achieves the maximum relative reduction rate in OBJ compared to the sub-optimal algorithm (original experiment record: 5.61%), indicating its most significant

advantage at this point. When $J = 30$, the DI-WOA achieves the minimum relative reduction rate in OBJ compared to the sub-optimal algorithm (original experiment record: 0.63%), indicating its smallest advantage at this point. The above comparison reflects the DI-WOA's advantage of maintaining the lowest unified bill cost at most points under congested scenarios, with a maximum unified bill cost relative saving rate of 5.61% and a minimum of 0.63%.

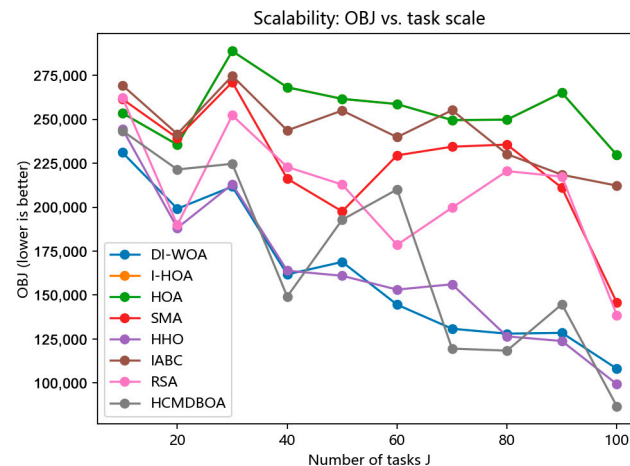


Figure 5. Curve of final objective function values under different task quantities.

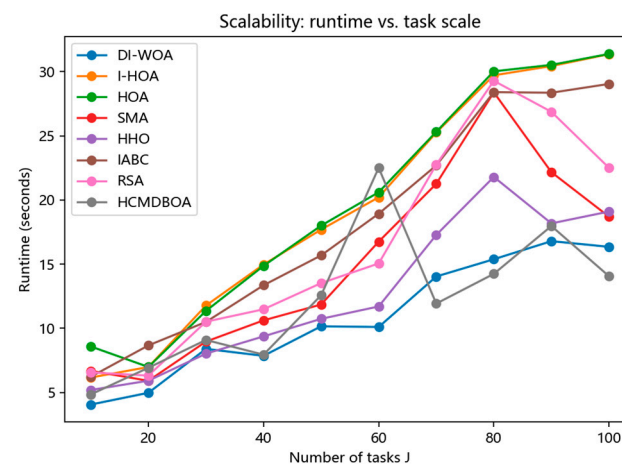


Figure 6. Curve of running time under different task quantities.

Figure 6 shows that the DI-WOA has a lower or second-lowest running time at all scale points, indicating that it can obtain high-quality solutions faster under the same iteration budget. When $J = 10$, the DI-WOA achieves the maximum relative reduction rate in running time compared to the sub-optimal algorithm (original experiment record: 16.28%), indicating its most significant advantage at this point. When $J = 40$, the DI-WOA achieves the minimum relative reduction rate in running time compared to the sub-optimal algorithm (original experiment record: 1.13%), indicating its smallest advantage at this point. The above comparison reflects the DI-WOA's advantage of maintaining the fastest convergence speed at most points under congested scenarios, with a maximum running time relative reduction rate of 16.28% and a minimum of 1.13%. This advantage mainly comes from two points:

- The discrete–continuous divide-and-conquer modeling directly solves the continuous frequency variables in the evaluation stage through a closed-form frequency step, reducing the search overhead of continuous dimensions;

- The enhanced Lagrange dual-rollback mechanism dynamically prices congested resources, enabling the search to migrate faster to low-default, low-cost regions, thereby improving convergence efficiency and stability.

In summary, the DI-WOA's scalability advantage mainly stems from the dimensionality reduction and rapid evaluation brought about by the closed-form frequency step, and the dynamic penalty pricing of congested lattices by the dual-rollback mechanism, enabling the algorithm to maintain good convergence efficiency and solution quality even as the task scale increases.

The results indicate that as the task scale increases, the DI-WOA achieves optimal or sub-optimal unified bill cost (OBJ) levels at the vast majority of task scales, with its advantage being most significant at $J = 60$; and it consistently maintains lower or the second-lowest running time at all scale points, with its advantage being most significant at $J = 10$, demonstrating significant dimensionality reduction and efficiency improvement characteristics. The following Table 4 quantitatively presents the relative performance indicators of the DI-WOA compared to other algorithms in terms of bill cost and running time under the scalability experiment:

Table 4. Summary table of relative performance indicators for the scalability experiment.

Relative Performance Indicator	Compared to Optimal Algorithm (%)	Compared to Sub-Optimal Algorithm (%)	Compared to Worst Algorithm (%)
Maximum Bill Cost Relative Saving Rate	0.00	5.61	52.96
Minimum Bill Cost Relative Saving Rate	−24.62	−8.87	14.19
Maximum Running Time Relative Reduction Rate	0.00	16.28	55.11
Minimum Running Time Relative Reduction Rate	−17.69	−4.61	28.69

4.2.2. Ablation Experiment and Result Analysis

The ablation experiment quantifies the contribution of two key mechanisms in the DI-WOA to solution quality improvement:

- Enhanced Lagrange dual-rollback (dual update), used for dynamic pricing of capacity soft constraint violations and guiding feasibility;
- Low-probability perturbation (perturbation) is used to maintain population diversity and reduce premature convergence risk. Using the basic WOA as the baseline, three variants are constructed: DI-WOA (dual + perturb), DI-WOA w/o dual (only perturbation retained), and DI-WOA w/o perturb (only dual-rollback retained). Ablation experiment results are shown in Figure 7 (best OBJ during iteration) and Figure 8 (final OBJ comparison).

Figure 7 shows that the baseline WOA declines slowly in early stages and stagnates relatively early; the WOA with perturbation introduced (DI-WOA w/o dual) stagnates later in its iteration trajectory compared to the baseline WOA and the WOA with dual feedback (DI-WOA w/o perturb), but converges to a lower best OBJ value. This indicates that the WOA with perturbation introduced (DI-WOA w/o dual) enhances early population diversity and explorability by superimposing small-probability Gaussian perturbations during iteration, reducing the probability of getting trapped in local optima. In contrast, the baseline WOA and the WOA with dual feedback (DI-WOA w/o perturb) may experience oscillations or premature convergence due to insufficient population diversity. The above experimental phenomena can preliminarily prove that the small-probability Gaussian perturbation mechanism can effectively reduce the algorithm's probability of falling into local optima; the complete DI-WOA only gradually begins to stagnate around iteration 60, while the other three variants stagnate before reaching iteration 20. Moreover, the complete DI-WOA converges to the lowest best OBJ value among all experimental variants. This

indicates that the “directional guidance” of dual feedback and the “coverage exploration” of the perturbation mechanism are complementary, achieving further exploration of infeasible regions in early stages while maintaining strong convergence guidance later, thus realizing the complete “exploration–guidance–convergence” iterative dynamics chain of the algorithm.

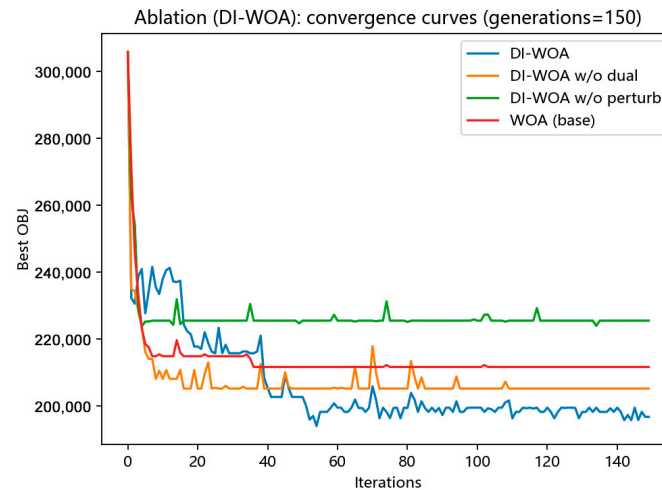


Figure 7. Curve of the best objective function value in the ablation study algorithms.

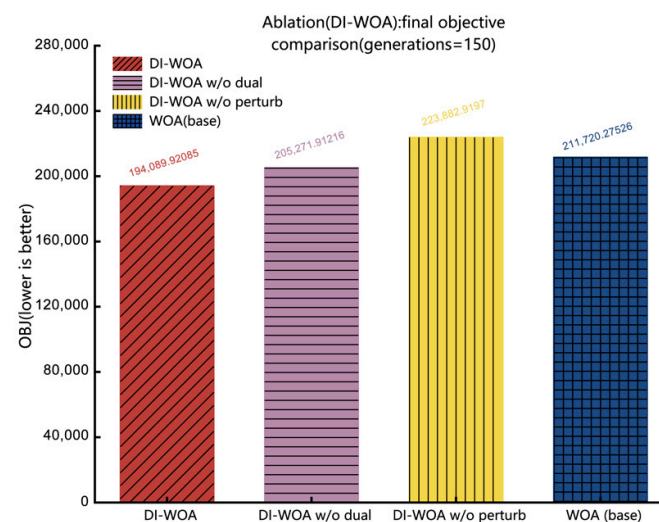


Figure 8. Final objective function value of the ablation study algorithms.

Figure 8 presents the final OBJ: the baseline WOA’s final OBJ is 211,720.28; the DI-WOA w/o dual’s final OBJ is 205,271.91 (3.05% lower than baseline), indicating that the perturbation mechanism alone can bring improvement by enhancing population diversity through superimposed small-probability Gaussian perturbations, making it easier to escape local optima; the DI-WOA w/o perturb’s final OBJ is 223,882.92 (5.74% higher than baseline), indicating that in the absence of increased diversity from small-probability Gaussian perturbations, dual feedback may cause the search to contract prematurely, reducing solution quality; and the complete DI-WOA’s final OBJ is 194,089.92, the lowest among all experimental algorithms, achieving unified bill cost relative saving rates of 8.33%, 5.45%, and 13.31% compared to the baseline, w/o dual, and w/o perturb, respectively. It can be observed that the combination of dual feedback and perturbation mechanisms yields better effects than either single mechanism, demonstrating that the “exploration–guidance–convergence” iterative dynamics chain formed by the complementarity of dual feedback’s “directional guidance” and the perturbation mechanism’s “coverage exploration” can reasonably bal-

ance the proportion of “exploration” and “exploitation” behaviors of the population at each iteration stage, significantly improving the WOA’s convergence performance and final solution quality on this scheduling model.

The experimental results indicate that both the “augmented Lagrangian dual-feedback” and “small-probability perturbation” mechanisms can improve solution quality, and their combined use yields the best effect. The formed “exploration → guidance → convergence” iterative dynamics chain can effectively balance explorability and exploitability, making the complete DI-WOA’s bill cost and penalty cost significantly superior to single-mechanism variants and the baseline algorithm. The following Table 5 quantitatively presents the relative performance indicators of the DI-WOA compared to other variant algorithms in terms of bill cost and penalty cost under the ablation experiment:

Table 5. Summary table of relative performance indicators for the ablation experiment.

Relative Performance Indicator	Compared to Baseline WOA (%)	Compared to DI-WOA w/o Dual (%)	Compared to Worst DI-WOA w/o Perturb (%)
Bill Cost Relative Saving Rate	8.33	5.45	13.31
Penalty Cost Relative Saving Rate	8.71	10.71	9.40

4.2.3. Robustness Experiment and Result Analysis

The robustness experiment simulates the impact of task request and resource capability fluctuations on scheduling strategies, evaluating the distribution stability of each algorithm under perturbed conditions and the fluctuation level of soft constraint penalty terms, to measure algorithm robustness. This experiment fixes $H = 4$, $T = 24$, and $J = 30$, runs five repetitions, with perturbation intensity set to 0.1. Perturbation simulates engineering uncertainty by applying $\pm 10\%$ relative random fluctuation to key parameters like task workload/resource supply; the final OBJ and soft constraint penalty term for each run were recorded, which are presented with box plots and summary tables (see Figures 9 and 10).

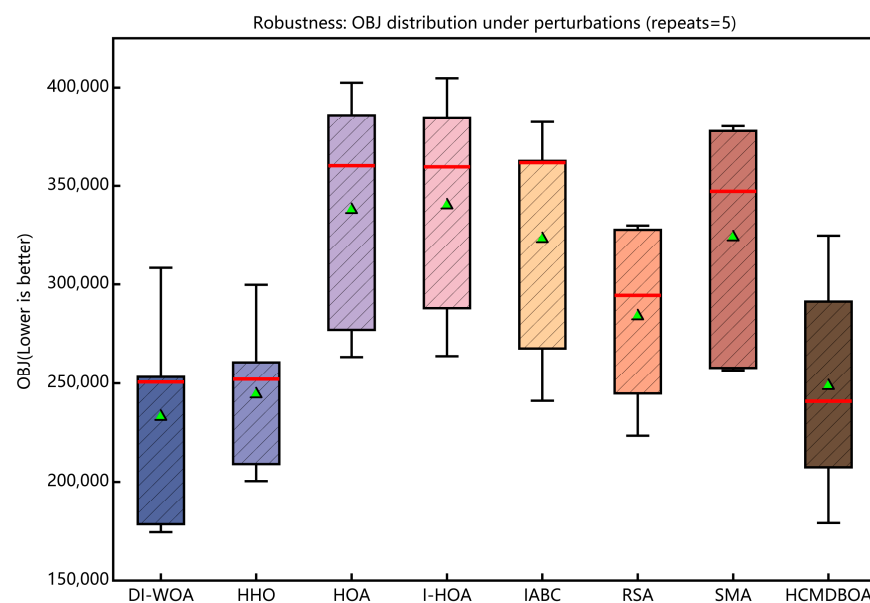


Figure 9. Box plots of OBJ values for each algorithm under perturbation conditions.

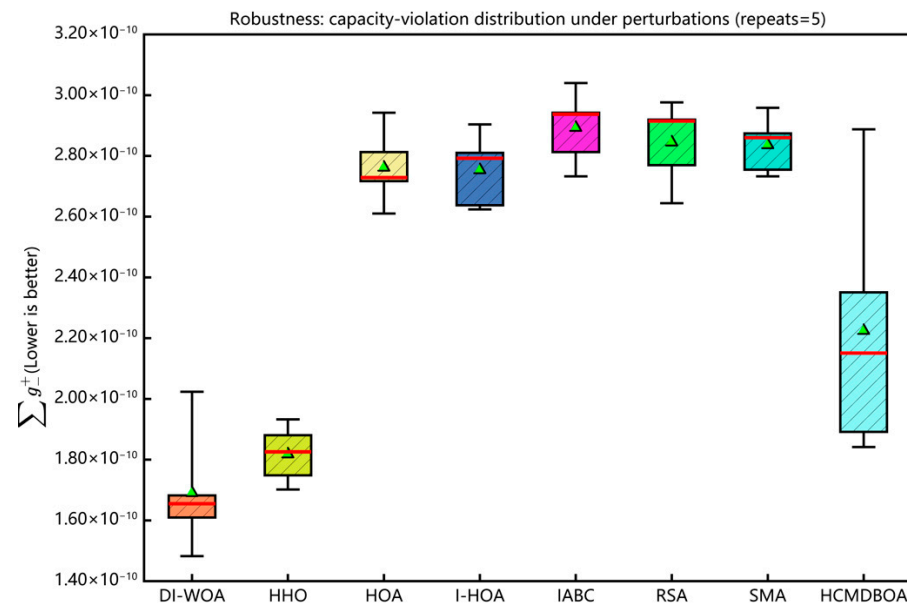


Figure 10. Box plots of $\sum g_{+}^{+}$ values for each algorithm under perturbation conditions.

As shown in Figure 9, the distributions of OBJ values for each algorithm under perturbed conditions show significant differences: a narrower box indicates smaller OBJ value fluctuations across runs, reflecting stronger perturbation resistance for bill cost; a lower median indicates the ability to obtain lower bill costs even under perturbations. The experimental results show that the DI-WOA's OBJ value has a relatively small IQR, approximately 74,831.2, indicating its OBJ output fluctuation can be maintained at the second-lowest level among all eight experimental algorithms. The DI-WOA's OBJ IQR is 44.74% higher than the optimal algorithm and 37.96% lower than the worst algorithm; the DI-WOA's OBJ median is relatively small, approximately 250,681, indicating its obtained bill cost under perturbation can be maintained at the second-lowest level among all eight experimental algorithms. The DI-WOA's median is 4.06% higher than the optimal algorithm and 30.71% lower than the worst algorithm.

As shown in Figure 10, the distribution of the $\sum g_{+}^{+}$ value for each algorithm under perturbation conditions differs significantly: a narrower bin indicates smaller fluctuations in the $\sum g_{+}^{+}$ value across runs, reflecting stronger disturbance resistance of the soft constraint violation amount; a lower median indicates that even under perturbation, a lower soft constraint violation amount can still be achieved. The experimental results show that the DI-WOA has the smallest corresponding IQR for $\sum g_{+}^{+}$, approximately 7.27596×10^{-12} , indicating that the DI-WOA can still maintain the lowest level of the $\sum g_{+}^{+}$ value output fluctuation among all eight experimental algorithms. The corresponding IQR for $\sum g_{+}^{+}$ to the DI-WOA is 23.81% lower than the sub-optimal algorithm and 41.72% lower than the worst algorithm; the median corresponding to the DI-WOA is the smallest, approximately 1.65528×10^{-10} , indicating that the DI-WOA can still achieve the lowest soft constraint violation amount among all eight experimental algorithms under perturbation. The median for $\sum g_{+}^{+}$ values of the DI-WOA are 9.34% lower than those of the sub-optimal algorithm and 43.65% lower than those of the worst algorithm.

Experimental results indicate that the DI-WOA's bill cost and its stability can both be maintained at sub-optimal levels, while its capacity soft constraint penalty cost and its stability are both optimal. This demonstrates that the DI-WOA exhibits exceptional robustness in perturbed environments, with more controllable output volatility. The following Table 6 quantitatively presents the relative performance indicators of the DI-

WOA compared to other algorithms in terms of bill cost and penalty cost under the robustness experiment:

Table 6. Summary table of relative performance indicators for the robustness experiment.

Relative Performance Indicator	Compared to Optimal Algorithm (%)	Compared to Sub-Optimal Algorithm (%)	Compared to Worst Algorithm (%)
Bill Cost Relative Saving Rate	−4.06	0.00	30.71
Bill Cost Stability Relative Increase Rate	−44.74	0.00	37.96
Penalty Cost Relative Saving Rate	0.00	9.34	43.65
Penalty Cost Stability Relative Increase Rate	0.00	23.81	41.72

4.2.4. Generalization Experiment and Result Analysis

The generalization experiment evaluates the adaptability of each algorithm under different resource sufficiency (util_target) and host scale (H) combinations, thereby testing the algorithm's cross-scenario generalization performance.

Generalization experiment settings: $T = 24$, $J = 30$, host number $H \in \{3, 4, 5\}$. Each scenario runs three repetitions, and the statistics are the mean OBJ. Load intensity constructs low/medium/high load scenarios via $\text{util_target} \in \{0.4, 0.6, 0.8\}$, where high load scenarios more easily trigger soft constraint penalties and deadline pressure. The mean OBJ results of each algorithm under different (H, util_target) combinations are shown in Figure 11, with relative percentage comparisons given in Tables 6 and 7.

Table 7. Percentage of superiority and inferiority of the mean objective value of the DI-WOA at $H = 3/4$.

Experimental Load Scenarios	DI-WOA Mean OBJ Above Optimal Algorithm (%)	DI-WOA Mean OBJ Below Sub-Optimal Algorithm (%)	DI-WOA Mean OBJ Below Worst Algorithm (%)
H = 3, util = 0.4	0.00	13.85	47.56
H = 3, util = 0.6	8.43	0.00	35.09
H = 3, util = 0.8	0.00	0.28	20.40
H = 4, util = 0.4	0.00	4.73	35.19
H = 4, util = 0.6	15.92	0.00	30.00
H = 4, util = 0.8	0.00	6.33	19.00

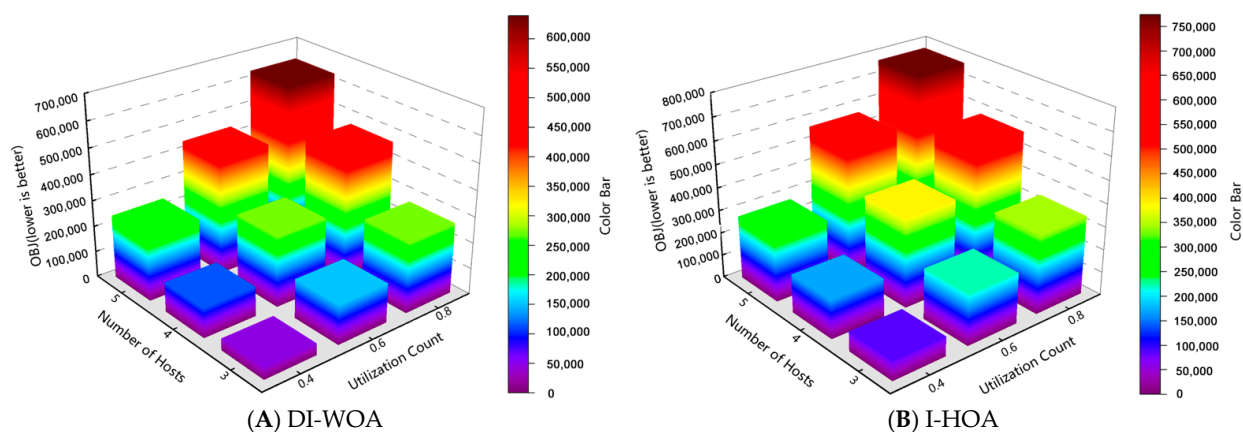


Figure 11. Cont.

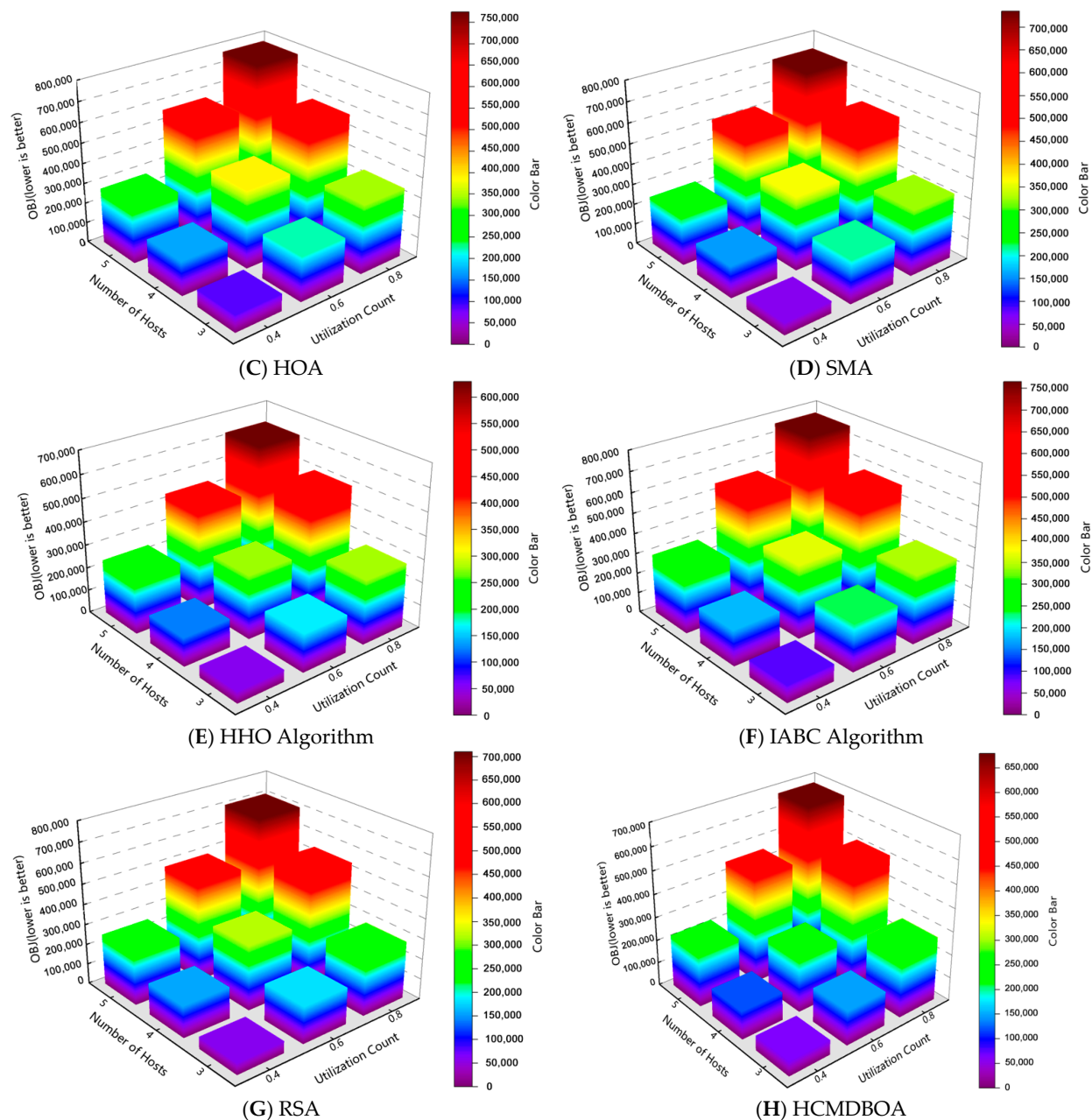


Figure 11. The mean OBJ of 7 experimental algorithms across scenarios.

As seen in Figure 11, as $util_target$ and H increase, the mean OBJ for all algorithms shows an overall upward trend. This trend mainly stems from:

- Increased load intensifies resource contention and raises delay/soft constraint penalty risk, causing mean OBJ to rise overall;
- Increased host scale expands the discrete decision space, and the concurrent increase in the number of interchangeable homogeneous hosts further multiplies symmetric redundant solutions, making the feasible region sparse and increasing the difficulty of searching for low mean OBJ solutions;
- Additionally, increased host scale leads to increased idle energy consumption, rising migration costs, and complexity in on-off strategies, further causing significant mean OBJ increase, greatly impacting bill cost. Thus, higher demands are placed on the algorithm's global search and feasibility guidance capabilities.

The DI-WOA's mean OBJ is lower than the other seven compared algorithms in $\frac{2}{3}$ experimental load scenarios for $H = 3$ and 4, and can maintain sub-optimal performance in the remaining $\frac{1}{3}$ of scenarios, demonstrating overall outstanding performance. The percentages by which the DI-WOA's mean OBJ is lower than the sub-optimal and worst algorithms are summarized in Table 7. The DI-WOA exhibits its greatest advantage at $H = 3$, $util = 0.4$, where its mean OBJ is 13.85% lower than the sub-optimal algorithm and 47.56% lower than the worst algorithm.

Table 8 gives relative differences between the DI-WOA and other algorithms at $H = 5$. In the $H = 5$, $util_target = 0.4$ scenario, the DI-WOA's mean OBJ is 8.51% higher than the optimal algorithm and 3.42% higher than the sub-optimal algorithm, but still 17.45% lower than the worst algorithm; in the $util_target = 0.6$ and 0.8 scenarios, the DI-WOA ties with the sub-optimal algorithm (0.00%), trailing the optimal algorithm by only 1.28% and 1.59%, respectively. Overall, the DI-WOA can still maintain sub-optimal performance in $\frac{2}{3}$ experimental load scenarios at $H = 5$, but its advantage margin converges somewhat compared to $H = 3/4$. This indicates that the DI-WOA has good adaptability to load variations, but to further unleash its potential at larger H scales in the future, it may be necessary to introduce dimensionality reduction mechanisms effective for large-scale host problems, such as:

- Introducing “hierarchical encoding” to decompose the complex scheduling decisions caused by increased host scale into multiple lower-dimensional decision levels for “divide and conquer”, thereby reducing the dimensionality of single decisions, making it easier for the algorithm to manage large-scale resources, and further reducing mean OBJ;
- “Symmetric group encoding” identifies and compresses all homogeneous hosts into a “symmetric group”, handled only as allocations to the “symmetric group” during decoding. This can effectively circumvent the emergence of numerous symmetric redundant solutions, preventing the algorithm from wasting computational resources on ineffective searches in these repeated regions, making it easier to search for low mean OBJ solutions.

Table 8. Percentage of superiority and inferiority of the mean objective value of the DI-WOA at $H = 5$.

Experimental Load Scenarios	DI-WOA Mean OBJ Above Optimal Algorithm (%)	DI-WOA Mean OBJ Above Sub-Optimal Algorithm (%)	DI-WOA Mean OBJ Below Worst Algorithm (%)
$H = 5$, $util = 0.4$	8.51	3.42	17.45
$H = 5$, $util = 0.6$	1.28	0.00	22.16
$H = 5$, $util = 0.8$	1.59	0.00	17.42

Based on Figure 11 and Tables 7 and 8, the DI-WOA achieves optimal or sub-optimal mean OBJ in all scenarios for $H = 3/4$ and remains competitive at $H = 5$, with gaps to the best algorithm within approximately 1–9% depending on the load level. This demonstrates that the DI-WOA possesses excellent cross-scenario generalization ability. The following Table 9 quantitatively presents the relative performance indicators of the DI-WOA compared to other algorithms in terms of average bill cost under the generalization experiment:

Table 9. Summary table of relative performance indicators for the generalization experiment.

Relative Performance Indicator	Compared to Optimal Algorithm (%)	Compared to Sub-Optimal Algorithm (%)	Compared to Worst Algorithm (%)
Maximum Average Bill Cost Relative Saving Rate	0.00	13.85	47.56
Minimum Average Bill Cost Relative Saving Rate	−8.51	−3.42	17.45

5. Conclusions and Future Work

This paper addresses the cost–efficiency trade-off and complex constraint-solving challenges in cloud server computing resource scheduling, proposing a unified monetized single-objective modeling and DI-WOA solution framework. This framework achieves dimensionality reduction via discrete–continuous divide-and-conquer and a closed-form frequency step; simultaneously, the three-layer structure of “discrete genes–real-valued encoding–decoder” better exploits the modeling structure and reduces search variable dimensionality, improving the algorithm’s ability to search for elite feasible solutions.

From a symmetry/asymmetry viewpoint, the unified monetization mechanism provides a scale-consistent objective for cross-scenario comparison, the encoding–decoding structure implicitly reduces redundant symmetric representations induced by interchangeable resources, and the enhanced Lagrange dual-rollback mechanism introduces beneficial asymmetry via adaptive congestion prices to guide feasibility restoration under soft constraints. The experimental results show that under baseline settings, the DI-WOA reduces the final OBJ by 8.33% compared to the baseline WOA; in multi-load generalization experiments, the DI-WOA’s mean OBJ is optimal in all scenarios for $H = 3/4$, leading the sub-optimal algorithm by up to 13.85%, verifying the effectiveness and stability of the proposed method under the unified bill cost objective. For large-scale H scenarios, hierarchical and symmetry-aware encoding (e.g., canonicalization over homogeneous host groups) and adaptive iteration budget/population size strategies can be utilized to control search difficulties arising from high-dimensional discrete space;

While the DI-WOA demonstrates stable performance on static instances and within limited-scale settings, it is imperative to acknowledge the inherent limitations of this framework. These limitations stem from the unpredictability and variability of real-world application scenarios, representing common shortcomings of heuristic algorithms. Specific limitations include:

- The unified monetization mechanism necessitates converting physical quantities of different dimensions into a single “bill cost,” which heavily relies on scenario-specific “conversion rates” and lacks universality for arbitrary contexts;
- The objective function modeling assumes static/known pricing signals (e.g., energy/carbon prices), making it difficult to adapt to scenarios with time-varying prices;
- The modeling presupposes offline instances with full prior knowledge of all task parameters (e.g., arrival times, deadlines, and workloads), rendering it unsuitable for volatile scenarios involving dynamically arriving or cancelled tasks.

Building upon the aforementioned three shortcomings, future work will focus on developing a general-purpose monetization calibration mechanism adaptable to the cost characteristics of diverse scenarios. Furthermore, it will explore extending offline scheduling to online horizons by integrating techniques from the reinforcement learning domain and neural networks for stochastic price prediction and dynamic task handling.

Author Contributions: Conceptualization, Y.K. and H.L.; methodology, Y.K.; software, Z.Z.; validation, Y.K., Z.Z. and H.L.; formal analysis, Z.Z.; investigation, Y.K.; resources, H.L.; data curation, Z.Z.; writing—original draft preparation, Y.K.; writing—review and editing, Z.Z. and H.L.; visualization, Z.Z.; supervision, H.L.; project administration, H.L.; funding acquisition, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: Hanshen Li was employed by the China Electric Power Planning & Engineering Institute. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Beloglazov, A.; Buyya, R.; Lee, Y.C.; Zomaya, A. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *Adv. Comput.* **2011**, *82*, 47–111.
2. Katal, A.; Kazemi, P.; Heidari, H.; Momeni, M. Energy efficiency in cloud computing data centers: A survey. *Clust. Comput.* **2023**, *26*, 1953–1980. [[CrossRef](#)]
3. Zhan, Z.H.; Liu, X.F.; Gong, Y.J.; Zhang, J.; Chung, H.S.H.; Li, Y. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv.* **2015**, *47*, 63. [[CrossRef](#)]
4. Kalra, M.K.; Singh, S. A review of metaheuristic scheduling techniques in cloud computing. *Egypt. Inform. J.* **2015**, *16*, 275–295. [[CrossRef](#)]
5. Sana, U.M.; Li, Z.L. Efficiency aware scheduling techniques in cloud computing: A descriptive literature review. *PeerJ Comput. Sci.* **2021**, *7*, e509. [[CrossRef](#)]
6. Ghribi, C.; Hadji, M.; Zeghlache, D. Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms. In Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Delft, The Netherlands, 13–16 May 2013; pp. 671–678.
7. Qiu, Y.; Jiang, C.; Wang, Y.; Ou, D.; Li, Y.; Wan, J. Energy aware virtual machine scheduling in data centers. *Energies* **2019**, *12*, 646. [[CrossRef](#)]
8. Kumar, M.; Sharma, S.C.; Goel, A.; Singh, S. A comprehensive survey for scheduling techniques in cloud computing. *J. Netw. Comput. Appl.* **2019**, *143*, 1–33. [[CrossRef](#)]
9. Aktan, M.N.; Bulut, H. Metaheuristic task scheduling algorithms for cloud computing environments. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6513. [[CrossRef](#)]
10. Vijaya, C.; Jeyanthi, P.; Kannan, P. Multi-objective meta-heuristic technique for energy efficient resource allocation in cloud computing. *Informatica* **2024**, *48*, 63–76. [[CrossRef](#)]
11. Dai, X.; Wang, J.M.; Bensaou, B. Energy-efficient virtual machines scheduling in multi-tenant data centers. *IEEE Trans. Cloud Comput.* **2015**, *4*, 210–221. [[CrossRef](#)]
12. Alourani, A.; Khalid, A.; Tahir, M.; Sardaraz, M. Energy efficient virtual machines placement in cloud datacenters using genetic algorithm and adaptive thresholds. *PLoS ONE* **2024**, *19*, e0296399. [[CrossRef](#)]
13. Elsedimy, E.I.; Herajy, M.; Abohashish, S.M.M. Energy and QoS-aware virtual machine placement approach for IaaS cloud datacenter. *Neural Comput. Appl.* **2025**, *37*, 2211–2237. [[CrossRef](#)]
14. Liu, B.; Chen, R.; Lin, W.; Wu, W.; Lin, J.; Li, K. Thermal-aware virtual machine placement based on multi-objective optimization. *J. Supercomput.* **2023**, *79*, 12563–12590. [[CrossRef](#)]
15. Yassa, S.; Chelouah, R.; Kadima, H.; Granado, B. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. *Sci. World J.* **2013**, *2013*, 350934. [[CrossRef](#)] [[PubMed](#)]
16. Abazari, F.; Analoui, M.; Takabi, H.; Fu, S. MOWS: Multi-objective workflow scheduling in cloud computing based on heuristic algorithm. *Simul. Model. Pract. Theory* **2019**, *93*, 119–132. [[CrossRef](#)]
17. Anwar, N.; Deng, H. A Hybrid Metaheuristic for Multi-Objective Scientific Workflow Scheduling in a Cloud Environment. *Appl. Sci.* **2018**, *8*, 538. [[CrossRef](#)]
18. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [[CrossRef](#)]
19. Sreenu, K.; Sreelatha, M. W-Scheduler: Whale optimization for task scheduling in cloud computing. *Clust. Comput.* **2019**, *22*, 1087–1098. [[CrossRef](#)]
20. Han, L.; Zhu, S.; Zhao, H.; He, Y. An enhanced whale optimization algorithm for task scheduling in edge computing environments. *Front. Big Data* **2024**, *7*, 1422546. [[CrossRef](#)] [[PubMed](#)]

21. Zhang, Y.; Wang, J. Enhanced whale optimization algorithm for task scheduling in cloud computing environments. *J. Eng. Appl. Sci.* **2024**, *71*, 121. [[CrossRef](#)]
22. Mohammadzadeh, A.; Chhabra, A.; Mirjalili, S.; Faraji, A. Use of whale optimization algorithm and its variants for cloud task scheduling: A review. In *Handbook of Whale Optimization Algorithm*; Academic Press: Cambridge, MA, USA, 2024; pp. 47–68.
23. Li, H.J.; Zheng, P.; Wang, T.T.; Wang, J.J.; Liu, T.M. A multi-objective task offloading based on BBO algorithm under deadline constrain in mobile edge computing. *Clust. Comput.* **2023**, *26*, 4051–4067. [[CrossRef](#)]
24. Li, H.J.; Liu, L.J.; Duan, X.L.; Li, H.Y.; Zheng, P.; Tang, L.B. Energy-efficient offloading based on hybrid bio-inspired algorithm for edge–cloud integrated computation. *Sustain. Comput. Inform. Syst.* **2024**, *42*, 100972. [[CrossRef](#)]
25. Shukri, S.E.; Al-Sayyed, R.; Hudaib, A.; Mirjalili, S. Enhanced multi-verse optimizer for task scheduling in cloud computing environments. *Expert Syst. Appl.* **2021**, *168*, 114230. [[CrossRef](#)]
26. Amer, D.A.; Attiya, G.; Zeidan, I.; Nasr, A.A. Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. *J. Supercomput.* **2022**, *78*, 2793–2818. [[CrossRef](#)]
27. Fatehi, S.; Motameni, H.; Barzegar, B.; Golsorkhtabaramiri, M. Energy aware multi objective algorithm for task scheduling on DVFS-enabled cloud datacenters using fuzzy NSGA-II. *Int. J. Nonlinear Anal. Appl.* **2021**, *12*, 2303–2331.

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.