

Article

Fog Task Scheduling Using Quality-Source-Driven Multi-Anchor Synchronized Search Algorithm

Haitao Xie ^{1,2}, Zhuo Luo ¹, Zhiwei Ye ^{1,2,*}, Wen Zhou ^{1,2}, Xianjing Zhou ³, Donglei Xu ⁴ and Mingming Zhao ⁴

¹ School of Computer Science and Artificial Intelligence, Hubei University of Technology, Wuhan 430068, China; 19991035@hbut.edu.cn (H.X.); 102401148@hbut.edu.cn (Z.L.); zw_mmwh@whu.edu.cn (W.Z.)

² Hubei Provincial Key Laboratory of Green Intelligent Computing Power Network, Hubei University of Technology, Wuhan 430068, China

³ Wuhan Zhuoer Information Technology Co., Ltd., Wuhan 430312, China; zxjzallcorp@163.com

⁴ Wuhan Fiberhome Technical Services Co., Ltd., Wuhan 430205, China; dlxu@fiberhome.com (D.X.); mmzhao@fiberhome.com (M.Z.)

* Correspondence: hgcsyzw@hbut.edu.cn

Abstract

Efficient task scheduling in heterogeneous IoT–Fog environments is challenging due to limited fog resources, diverse task demands, and conflicting QoS objectives. This paper proposes ASQS, a Quality-Source-driven Multi-Anchor Synchronized Search algorithm for IoT–Fog task scheduling. ASQS is biomimetically motivated by collective search behaviors in natural systems, where distributed exploration, collective memory, and probabilistic cooperation support an exploration–exploitation balance. Specifically, ASQS constructs quality layers from candidate schedules, extracts representative quality-source anchors, and reuses them through an ACO-inspired probabilistic synchronization mechanism, thereby improving the utilization of high-quality historical search information. FNO-based search and Lévy-flight perturbation are further incorporated to enhance directional guidance and long-range exploration. Experiments on 33 benchmark functions, ablation studies, sensitivity analysis, standard fog scheduling scenarios, and large-scale task-intensive scenarios were conducted to evaluate ASQS. The results show that ASQS achieves competitive optimization accuracy, stable convergence, and superior comprehensive scheduling performance in terms of fitness, makespan, latency, load balance, and constraint handling. In particular, the large-scale experiment with 100 fog nodes and up to 8000 IoT tasks verifies the scalability of ASQS under heavy workload pressure. Statistical tests further confirm the reliability of the observed improvements. These results demonstrate that ASQS is an effective, scalable, and biomimetically motivated optimizer for IoT–Fog task scheduling.

Keywords: IoT–Fog task scheduling; quality-source anchors; multi-anchor synchronization; metaheuristic algorithm; large-scale scheduling



Academic Editor: Heming Jia

Received: 4 May 2026

Revised: 24 May 2026

Accepted: 26 May 2026

Published: 3 June 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

The rapid proliferation of IoT applications has generated a large number of latency-sensitive task requests from edge devices. Sole reliance on remote cloud servers is often inefficient, as long-distance data transmission may introduce substantial latency and bandwidth pressure [1,2]. Fog computing alleviates this limitation by deploying computing resources closer to IoT devices; however, the constrained and heterogeneous nature of fog resources makes task scheduling a critical challenge. Inefficient scheduling may result in

node overload, prolonged completion time, and degraded quality of service, highlighting the necessity of efficient IoT–Fog task scheduling in latency-sensitive environments [3,4]. Figure 1 illustrates a typical three-layer fog computing architecture, where IoT devices generate task requests, fog nodes provide nearby computing resources, and cloud servers offer centralized support for large-scale processing.

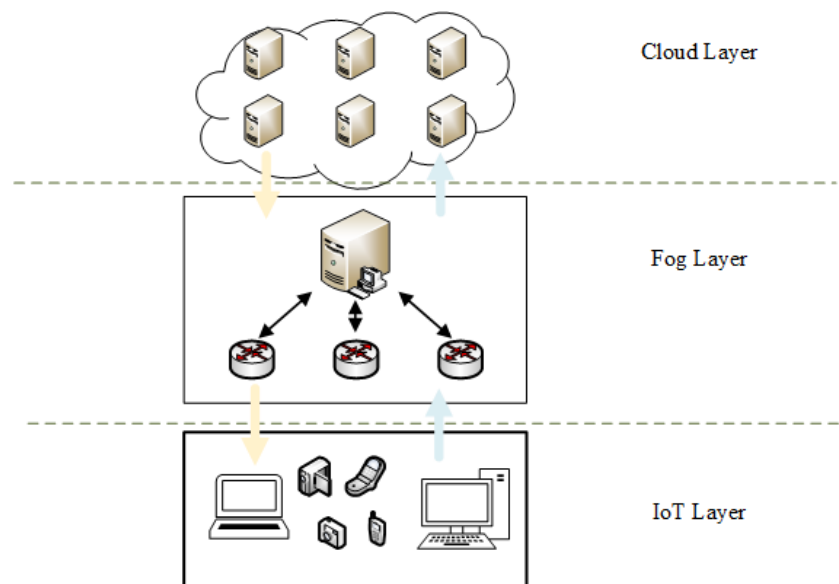


Figure 1. Typical fog computing architecture consisting of the IoT layer, fog layer, and cloud layer, where the arrows indicate the direction of data flow and communication between different layers.

Fog task scheduling is challenging because both resources and workloads are highly heterogeneous. Fog nodes, including gateways, routers, base stations, and edge servers, usually differ in computing capacity, memory, bandwidth, energy consumption, and availability, while IoT tasks have diverse requirements in CPU cycles, input data size, memory demand, priority, and deadlines. The scheduler must assign each task to a suitable fog node under resource and deadline constraints, forming a discrete task–node assignment problem. Moreover, multiple QoS objectives, such as makespan, response latency, energy consumption, load balance, resource utilization, and task completion rate, are often conflicting and must be optimized simultaneously [5,6]. Existing time-aware and multi-objective scheduling studies have attempted to improve deadline satisfaction and global scheduling quality [5]; however, reliability, dependability, and dynamic decision-making remain important concerns in practical IoT–Fog systems [7–9]. Therefore, fog task scheduling is generally treated as an NP-hard combinatorial optimization problem.

Recent studies have increasingly moved beyond conventional heuristic and meta-heuristic schedulers toward hybrid, learning-based, and adaptive optimization frameworks for IoT–Fog task scheduling, including deep reinforcement learning and fuzzy deep reinforcement learning-based scheduling methods [10,11]. For example, Heirati et al. integrated deep learning, reinforcement learning, and metaheuristic optimization to improve energy efficiency, task completion time, and QoS [12]. Yu et al. developed NF-MORL, which combines Takagi–Sugeno fuzzy inference with multi-objective actor–critic learning to optimize completion time, energy consumption, cost, and fault tolerance under uncertainty [13]. Mindil et al. proposed QBDS, a quantum-inspired scheduler that uses priority-aware ranking, sinusoidal biasing, and penalty-aware multi-objective evaluation to improve makespan, energy, cost, load balancing, response time, and scalability [1].

Despite extensive research on IoT–Fog scheduling, several gaps remain unresolved. First, heterogeneous tasks, resource-constrained fog nodes, conflicting QoS objectives,

and dynamic system states still make task-node assignment a difficult optimization problem [14–17]. Second, although heuristic, metaheuristic, and learning-based schedulers have achieved improvements, population-based search remains sensitive to parameter settings and is still prone to premature convergence and exploration–exploitation imbalance [12,13,18,19]. Third, the main weakness of many hybrid metaheuristic schedulers is not the absence of search operators but the absence of a structured mechanism for quality-source management: high-quality search experience is often generated but not explicitly extracted, archived, and reused as multi-source guidance across promising regions [12,18,19]. Fourth, many studies still under-specify objective aggregation, penalty-based constraint handling, parameter sensitivity, scalability behavior, and fair comparison protocols, thereby limiting reproducibility and the reliability of performance claims [16,17,20,21].

Motivated by these observations, this paper proposes ASQS, a quality-source-driven multi-anchor synchronized search algorithm for resource-constrained IoT–Fog task scheduling. The design of ASQS is closely related to biomimetic optimization principles observed in natural collective systems, such as ant colonies, bird flocks, and animal foraging groups, where distributed exploration, collective memory, and probabilistic cooperation are commonly used to balance exploration and exploitation. Unlike conventional hybrid metaheuristics that mainly accumulate multiple operators or rely heavily on a single global-best solution, ASQS shifts the focus to quality-source management. Specifically, high-quality search experience is extracted from layered populations, archived as reusable anchors, and probabilistically reused through ACO-inspired selection and anchor-seeding to provide multi-source guidance over multiple promising regions. Combined with FNO-based directional search and Lévy-flight perturbation, which supports occasional long-range exploration consistent with biological foraging behavior, the proposed framework alleviates excessive single-elite dependence and forms a biomimetically motivated search process for IoT–Fog scheduling.

The main contributions of this paper are summarized as follows:

1. Establishment of a resource-constrained IoT–Fog scheduling model with explicit task-node encoding, QoS-aware objective aggregation, and penalty-based treatment of resource and deadline violations.
2. Introduction of a biomimetically motivated quality-source management mechanism that extracts high-quality solutions from layered populations and organizes them as reusable anchors, reflecting collective memory and multi-source experience reuse in natural systems.
3. Design of an ACO-inspired multi-anchor synchronized search strategy that probabilistically reuses archived anchors as multi-source guidance, mimicking indirect information sharing and cooperative path reinforcement while mitigating single-elite dependence.
4. Comprehensive validation based on benchmark-function tests, IoT–Fog scheduling simulations, algorithm comparisons, ablation studies, sensitivity analysis, scalability evaluation, and statistical tests.

The remainder of this paper is organized as follows: Section 2 reviews related work on IoT–Fog task scheduling, metaheuristic scheduling methods, and search experience reuse. Section 3 introduces the component roles and design motivation of the proposed ASQS. Section 4 presents the ASQS algorithm in detail, including solution encoding, fitness evaluation, search operators, quality-source archive construction, and the overall procedure. Section 5 reports the experimental studies, including benchmark-function comparisons, sensitivity analysis, ablation experiments, and IoT–Fog task scheduling experiments. Finally, Section 6 concludes this paper and discusses future research directions.

2. Related Works

2.1. IoT–Fog Task Scheduling Models and QoS Objectives

IoT–Fog task scheduling has been widely studied as a resource-constrained task-node assignment problem, where heterogeneous IoT tasks are allocated to heterogeneous fog nodes under capacity, latency, energy, and deadline constraints [22–25]. Existing studies have considered various QoS objectives, including makespan, response time, energy consumption, execution cost, load balancing, resource utilization, and task completion rate [23,26]. Time-aware, deadline-constrained, and multi-objective formulations have improved the applicability of fog scheduling in latency-sensitive environments [23,26]. However, many studies still provide limited details on continuous-to-discrete task-node encoding, normalization of heterogeneous QoS metrics, weighted objective aggregation, infeasible assignment repair, and penalty-based handling of resource and deadline violations, which may reduce reproducibility and weaken fair comparison.

2.2. Heuristic, Metaheuristic, and Learning-Based Scheduling Methods

To solve the NP-hard IoT–Fog scheduling problem, heuristic, metaheuristic, and learning-based methods have been extensively investigated [26,27]. Heuristic methods usually offer low computational cost but may be trapped in locally optimal scheduling decisions. Population-based metaheuristics improve global search by maintaining multiple candidate solutions and have been applied to balance latency, energy, cost, and resource utilization [23,27]. More recently, reinforcement learning, deep learning, neuro-fuzzy learning, and adaptive scheduling frameworks have been introduced to enhance decision-making in dynamic fog environments [27,28]. Despite these advances, metaheuristic schedulers remain sensitive to parameter settings and may suffer from premature convergence, while learning-based methods often require careful reward design, sufficient training data, and additional computational overhead.

2.3. Hybrid Metaheuristics and Search Experience Reuse

Hybrid metaheuristics have become a common direction for improving IoT–Fog scheduling performance by combining complementary mechanisms such as global exploration, local exploitation, elite learning, chaotic initialization, Lévy-flight perturbation, adaptive parameter control, and pheromone-inspired selection [27,29]. These designs can enhance search capability, but many of them remain operator-centric, where performance improvement mainly depends on adding or modifying search operators. In such frameworks, high-quality solutions generated during evolution are often used only implicitly through the current best individual or elite replacement, rather than being explicitly extracted, archived, and reused as structured guidance sources. Therefore, the systematic management of quality sources and their reuse as multi-source guidance remain insufficiently explored in hybrid scheduling optimization.

2.4. Summary and Research Positioning

Overall, existing IoT–Fog scheduling studies have improved QoS-aware task allocation through advanced modeling, metaheuristic optimization, and learning-based decision-making [22,23,27]. However, three limitations remain evident: the optimization formulation is not always sufficiently transparent, population-based search may still be unstable in complex discrete scheduling spaces, and hybrid metaheuristics often lack explicit mechanisms for preserving and reusing high-quality search experience. To address these issues, ASQS is positioned as a quality-source-driven scheduling optimizer that combines explicit scheduling formulation with anchor-based experience preservation and multi-anchor synchronized

search, aiming to improve search stability, diversity maintenance, and reproducibility in resource-constrained IoT–Fog scheduling.

3. Component Roles and Design Motivation of ASQS

The proposed ASQS framework is not a direct combination of several standalone metaheuristic algorithms. Instead, it reorganizes selected search principles into a quality-source-driven multi-anchor search framework. Specifically, the hierarchical population organization is used to extract quality sources from different fitness layers, the FNO-based operator provides directional search guidance, the Lévy-flight mechanism enhances escape ability, and the ACO-inspired rule is redesigned as a probabilistic synchronization strategy among anchors. In addition, the cooperative search principle, inspired by cooperative metaheuristic algorithms [30], is incorporated to coordinate information exchange among quality-source anchors and candidate schedules, thereby supporting collaborative exploration and exploitation within the unified ASQS framework.

Table 1 summarizes the role of each component and clarifies whether it is newly proposed or adapted from existing techniques.

Table 1. Summary of the main components in ASQS.

Component	Basis	Function	Effect	Type
Hierarchical organization	HRO-inspired	Quality-layer grouping	Structured diversity	Adapted
FNO-guided search	FNO	Better/worse reference update	Global-local balance	Adapted
Lévy escape perturbation	Lévy flight	Conditional long-jump perturbation	Exploration	Adapted
Quality-source archive	Proposed	Multi-layer elite storage	Multi-center guidance	Proposed
Anchor seeding	Proposed	Anchor-neighborhood generation	Local refinement	Proposed
ACO-inspired synchronization	ACO-inspired	Probabilistic anchor sharing	Cooperative convergence	Hybrid

3.1. Motivation of Quality-Source-Driven Multi-Anchor Search

Inspired by the hierarchical population organization mechanism of Hybrid Rice Optimization (HRO) [31], ASQS ranks candidate schedules by fitness and organizes them into different quality layers. This layered organization enables ASQS to extract representative quality-source anchors from different fitness regions rather than relying only on top-ranked solutions. Table 2 summarizes the role of each quality layer in anchor construction, where high-quality, medium-quality, and low-quality layers respectively provide exploitation-oriented, stability-oriented, and diversity-oriented sources for the subsequent multi-anchor synchronized search.

Table 2. Roles of different quality layers in anchor construction.

Layer in ASQS	Fitness-Based Criterion	Function in Anchor Construction
High-quality layer	Top-ranked	Provides exploitation-oriented quality sources.
Medium-quality layer	Middle-ranked	Provides balanced quality sources for maintaining search stability.
Low-quality layer	Bottom-ranked	Provides diversity-oriented sources to enlarge the search scope.

3.2. Roles of Adapted Search Operators

The global search strategy of ASQS draws on the directional guidance concept of the Farthest Better and Nearest Worse Optimizer (FNO), which relies on two contrastive references: the farthest better solution and the nearest worse solution [32]. In the global search stage, the farthest better reference encourages candidate schedules to move toward remote higher-quality regions, while the nearest worse reference helps reduce movements toward nearby inferior regions. This fitness-contrast-based guidance enables ASQS to

extend the search scope in a more directed manner and is coordinated with the quality-source anchor archive during candidate updates.

ASQS adopts a probability-guided multi-anchor synchronization strategy motivated by the selection and feedback principles of Ant Colony Optimization (ACO) [33]. In this strategy, quality-source anchors are selected according to probabilities determined by their fitness-based importance and layer information. As a result, high-quality anchors provide stronger guidance for candidate updates, while medium- and low-quality anchors are also involved to preserve search diversity. This probability-guided synchronization helps coordinate information sharing among multiple anchors within the ASQS search process.

ASQS employs a Lévy-flight-inspired escape perturbation motivated by the heavy-tailed search behavior of Lévy flight [34]. This behavior produces frequent short-range perturbations together with occasional long-range jumps, enabling candidate schedules to move out of restricted search regions when the search progress slows down. The perturbation is applied conditionally within the ASQS search process and works in coordination with the quality-source anchor archive to help maintain exploration diversity.

4. Proposed ASQS Algorithm

4.1. Overall Framework of ASQS

ASQS is organized as a quality-source-guided closed-loop search framework. At each iteration, FNO-based directional search and fitness-adaptive Lévy-flight escape first expand the current population and form an intermediate candidate pool. Quality-source anchors are then extracted from this pool across different fitness layers to construct the anchor archive. The archive provides multi-anchor guidance for the ACO-inspired synchronization module, enabling candidate updates to exploit multiple promising scheduling regions instead of a single best solution. The synchronized candidates are finally evaluated and filtered through greedy selection to form the next population. Thus, the main components of ASQS are integrated as cooperative functional operators within a unified iterative procedure, rather than as independent optimizers.

4.2. Solution Encoding and Fitness Evaluation

To apply ASQS to fog task scheduling, each continuous candidate generated by the search operators must be decoded into a discrete task-node assignment before evaluation. This subsection defines the solution representation, decoding rule, scheduling metrics, constraint handling strategy, and greedy selection criterion used in ASQS.

In ASQS, the p th candidate solution is represented by a continuous vector $X_p = [x_{p,1}, x_{p,2}, \dots, x_{p,n}]$, where n is the number of IoT tasks and each dimension corresponds to one task. Before evaluation, X_p is decoded into a discrete assignment vector $A_p = [a_{p,1}, a_{p,2}, \dots, a_{p,n}]$, where $a_{p,i} \in \{1, 2, \dots, m\}$ denotes the fog node selected for task T_i , and m is the number of fog nodes. Each dimension is first clipped into $[0, 1]$, and the discrete node index is obtained as

$$x_{p,i} = \min(1, \max(0, x_{p,i})), \quad a_{p,i} = \min(m, \max(1, \lceil x_{p,i} \cdot m \rceil)). \quad (1)$$

This decoding rule ensures that each task is assigned to exactly one fog node.

For a decoded schedule A_p , if task T_i is assigned to fog node $F_{a_{p,i}}$, its transmission time, execution time, and basic processing time are calculated as

$$TT_{p,i} = \frac{D_i}{B_{a_{p,i}}}, \quad ET_{p,i} = \frac{L_i}{C_{a_{p,i}}}, \quad PT_{p,i} = TT_{p,i} + ET_{p,i}, \quad (2)$$

where D_i and L_i denote the input data size and computational workload of task T_i , respectively, and $B_{a_{p,i}}$ and $C_{a_{p,i}}$ denote the bandwidth and computing capacity of the assigned fog node. Tasks assigned to the same fog node are executed sequentially according to the task index order. Let Q_j denote the accumulated finish time of fog node F_j , with $Q_j = 0$ initially. The final completion time of task T_i and the corresponding node update are defined as

$$FT_{p,i} = Q_{a_{p,i}} + PT_{p,i}, \quad Q_{a_{p,i}} \leftarrow FT_{p,i}. \quad (3)$$

Based on the finish-time evaluation, the makespan, average service latency, and energy consumption of schedule A_p are calculated as

$$MS_p = \max_{1 \leq j \leq m} Q_j, \quad LAT_p = \frac{1}{n} \sum_{i=1}^n FT_{p,i}, \quad E_p = \sum_{i=1}^n P_{a_{p,i}} \cdot ET_{p,i}, \quad (4)$$

where $P_{a_{p,i}}$ is the power consumption rate of the assigned fog node. The latency is computed using $FT_{p,i}$ so that both processing time and waiting time are considered.

In addition to the above objective-related metrics, the deadline satisfaction rate and load-balance index are reported to evaluate schedule feasibility and workload distribution. The deadline satisfaction rate is defined as

$$DSR_p = \frac{1}{n} \sum_{i=1}^n \mathbb{I}(FT_{p,i} \leq DL_i), \quad (5)$$

where $\mathbb{I}(\cdot)$ is an indicator function that returns 1 if the condition is satisfied and 0 otherwise. A larger DSR_p indicates that a higher proportion of tasks are completed before their deadlines.

The load-balance index is calculated using the standard deviation of the final accumulated finish times of fog nodes:

$$LB_p = \sqrt{\frac{1}{m-1} \sum_{j=1}^m (Q_j - \bar{Q})^2}, \quad \bar{Q} = \frac{1}{m} \sum_{j=1}^m Q_j, \quad (6)$$

where Q_j denotes the final accumulated finish time of fog node F_j . A smaller LB_p indicates a more balanced workload distribution among fog nodes.

Since these metrics have different scales, they are normalized before aggregation:

$$MS_{p,norm} = \frac{MS_p}{MS_{ref}}, \quad E_{p,norm} = \frac{E_p}{E_{ref}}, \quad LAT_{p,norm} = \frac{LAT_p}{LAT_{ref}}. \quad (7)$$

In the simulation, $MS_{ref} = 2 \cdot \max(DL_i)$, $E_{ref} = (\sum_{j=1}^m P_j) \cdot \max(DL_i)$, and $LAT_{ref} = \max(DL_i)$. These reference values are fixed within each scheduling scenario and are used only for scale normalization. The normalized objective is defined as

$$F_{obj}(A_p) = w_1 MS_{p,norm} + w_2 E_{p,norm} + w_3 LAT_{p,norm}, \quad (8)$$

where $w_1 + w_2 + w_3 = 1$ and $w_1, w_2, w_3 \geq 0$.

The resource capacity violation and deadline violation are defined as

$$V_{cap}(A_p) = \sum_{j=1}^m \max \left(0, \sum_{i:a_{p,i}=j} r_i - R_j \right), \quad (9)$$

and

$$V_{ddl}(A_p) = \sum_{i=1}^n \max(0, FT_{p,i} - DL_i), \quad (10)$$

where r_i is the resource demand of task T_i , R_j is the resource capacity of fog node F_j , and DL_i is the deadline of task T_i .

For reporting purposes, the overall constraint violation is calculated as

$$V_{total}(A_p) = V_{cap}(A_p) + V_{ddl}(A_p). \quad (11)$$

The final fitness value is then given by

$$Fitness(A_p) = F_{obj}(A_p) + \lambda_1 V_{cap}(A_p) + \lambda_2 V_{ddl}(A_p), \quad (12)$$

where λ_1 and λ_2 are penalty coefficients. A smaller fitness value indicates a better scheduling solution.

To further improve feasibility, a repair operation is applied when capacity violations occur. For an overloaded fog node, the task with the largest resource demand is migrated to a node with the smallest current resource load that can satisfy the capacity constraint. If no feasible node exists, the task is moved to the node with the smallest current resource load to reduce the overload degree. If the repair process cannot fully eliminate the violation within the maximum number of attempts, the remaining violation is still penalized in the fitness function.

After a new candidate is generated by an ASQS update operator, it is decoded, repaired if necessary, and evaluated using the above fitness function. The greedy selection criterion is defined as

$$X_p \leftarrow \begin{cases} X_p^{new}, & Fitness(A_p^{new}) < Fitness(A_p^{old}), \\ X_p^{old}, & \text{otherwise.} \end{cases} \quad (13)$$

where A_p^{new} and A_p^{old} are the decoded assignment vectors corresponding to X_p^{new} and X_p^{old} , respectively.

4.3. FNO-Based Search Operator

The FNO-based operator is used in ASQS to provide directional search guidance for candidate schedules [32]. For each candidate X_p , the farthest better solution X_p^{FB} and the nearest worse solution X_p^{NW} are identified according to fitness and distance. Specifically, X_p^{FB} denotes the better solution with the largest distance from X_p , while X_p^{NW} denotes the worse solution with the smallest distance from X_p .

In ASQS, the FNO-based directional update is implemented in a velocity-assisted form:

$$V_p^{t+1} = wV_p^t + c_1r_1 \odot (X_p^{FB} - X_p^t) + c_2r_2 \odot (X_p^t - X_p^{NW}) + c_3r_3 \odot (X_{best}^t - X_p^t), \quad (14)$$

$$X_p^{t+1} = X_p^t + V_p^{t+1}. \quad (15)$$

where V_p^t is the velocity vector, X_{best}^t is the current global-best solution, r_1 , r_2 , and r_3 are random vectors sampled from $[0, 1]$, and $\odot$ denotes element-wise multiplication. The inertia weight is set to $w = 0.9 - 0.5 \times ratio$, where $ratio = t/T$, and the acceleration coefficients are set to $c_1 = c_2 = c_3 = 1.5$. The initial velocity is $V_0 = 0$.

The term associated with X_p^{FB} guides the candidate toward promising distant regions, while the term associated with X_p^{NW} helps avoid inferior neighboring regions. The global-best term further provides convergence guidance. In ASQS, this operator is not used as an independent optimizer but serves as a directional update component coordinated by the quality-source anchor archive.

4.4. Lévy-Flight Escape Mechanism

To enhance the ability of ASQS to escape from local optima, a Lévy-flight escape mechanism is incorporated after the FNO-based directional search. For each candidate, the escape operation is adaptively determined by an escape energy that combines its relative fitness gap with the remaining iteration ratio:

$$E_p^t = \frac{|Fitness(A_{best}^t) - Fitness(A_p^t)|}{|Fitness(A_{best}^t)| + \epsilon} \left(1 - \frac{t}{T}\right), \quad (16)$$

where $Fitness(A_p^t)$ is the fitness value of the decoded schedule corresponding to X_p^t , $Fitness(A_{best}^t)$ is the current best fitness value, T is the maximum number of iterations, and ϵ is a small positive constant used to avoid division by zero.

When $E_p^t \geq 1$, the p th candidate solution is perturbed by Lévy flight:

$$X_p^{escape} = X_p^t + \lambda \cdot \text{Levy}(\beta) \odot (X_p^t - X_{best}^t), \quad (17)$$

where $\lambda = 0.01$ is the perturbation scale, $\text{Levy}(\beta)$ denotes a Lévy random step with the distribution parameter $\beta = 1.5$, $\odot$ represents element-wise multiplication, and X_{best}^t is the current best candidate solution. The threshold is fixed to 1 so that Lévy perturbation is mainly applied to candidates with a sufficiently large relative fitness gap, especially in the early search stage. After the escape operation, the generated candidate is clipped into the continuous search range, decoded into a discrete task-node assignment, repaired if necessary, and evaluated using the fitness function.

4.5. Quality-Source Extraction and Anchor Archive Construction

After the three-layer FNO-based search and Lévy-flight perturbation, the updated population contains valuable search information distributed across different quality layers. Relying only on the global-best solution may cause excessive concentration around a single region and increase the risk of premature convergence. Therefore, ASQS extracts representative elite solutions from multiple layers and organizes them into a quality-source anchor archive. These anchors act as reusable guidance centers for the subsequent ACO-inspired multi-anchor synchronized search.

Let $\mathcal{P}_t^{(l)}$ denote the l th quality layer of the population at iteration t , where $l = 1, 2, 3$ correspond to the high-quality, medium-quality, and low-quality layers, respectively. After the search and perturbation stages, a subset of representative solutions is selected from each layer according to fitness ranking. The anchor subset extracted from layer l is defined as

$$A_t^{(l)} = \text{TopK}(\mathcal{P}_t^{(l)}, K_l), \quad l = 1, 2, 3, \quad (18)$$

where K_l is the number of anchors selected from layer l , and $\text{TopK}(\cdot)$ returns the top-ranked candidates with smaller fitness values. Compared with using only the best individual, this layer-wise extraction strategy preserves multiple promising directions and allows ASQS to reuse high-quality information discovered by different parts of the population.

The global anchor archive is constructed by merging the anchor subsets from all quality layers:

$$A_t = A_t^{(1)} \cup A_t^{(2)} \cup A_t^{(3)}. \quad (19)$$

Here, anchors from the high-quality layer mainly provide exploitation-oriented guidance, anchors from the medium-quality layer support a balanced search transition, and anchors from the low-quality layer preserve alternative directions for diver-

sity. Thus, the archive represents a multi-source guidance structure rather than a single elite solution.

To determine the contribution of different anchors in the subsequent synchronized search, each anchor is assigned a probability according to both its fitness ranking and its layer source. Let $M_t = |A_t|$ be the number of anchors in the archive and $rank(k)$ be the fitness rank of the k th anchor, where $rank(k) = 1$ indicates the best anchor. The ranking weight is defined as

$$w_r(k) = \frac{1}{rank(k)}. \quad (20)$$

In addition, the layer-dependent weight is defined as

$$w_l(k) = \begin{cases} \omega_1, & A_k \in A_t^{(1)}, \\ \omega_2, & A_k \in A_t^{(2)}, \\ \omega_3, & A_k \in A_t^{(3)}, \end{cases} \quad (21)$$

where ω_1 , ω_2 , and ω_3 control the relative influence of anchors from different layers. In this study, $\omega_1 = 1.0$, $\omega_2 = 0.7$, and $\omega_3 = 0.4$. In general, high-quality anchors are assigned larger weights to enhance exploitation, while medium- and low-layer anchors retain nonzero probabilities to maintain diversity.

The final selection probability of the k th anchor is calculated as

$$\pi_k = \frac{w_r(k)w_l(k)}{\sum_{j=1}^{M_t} w_r(j)w_l(j)}. \quad (22)$$

This probability design enables ASQS to prefer high-quality anchors while still allowing other representative anchors to participate in the search. The archive is updated at each iteration after the layered search and perturbation stages so that the synchronized search can be guided by the latest quality sources discovered by the population.

4.6. Anchor-Seeding and ACO-Inspired Multi-Anchor Synchronized Search

ASQS introduces an ACO-inspired multi-anchor synchronization strategy to share useful scheduling patterns among candidate solutions. The positive-feedback and probabilistic selection principles of ACO are adapted to anchor-level guidance rather than edge-level path construction. In classical ACO, paths with stronger pheromone information are more likely to be selected and further reinforced. In ASQS, this idea is transferred to the quality-source anchor archive: anchors with better fitness ranking and stronger layer importance are assigned higher selection probabilities and are therefore more likely to serve as guidance sources for subsequent candidate generation.

Anchor seeding provides the mechanism by which the information stored in the archive is injected back into the population. Instead of forcing all individuals to follow the current global-best solution, each individual probabilistically selects one anchor from the archive constructed in Section 4.5. The selected anchor is used as a seed center for generating a synchronized candidate. Therefore, anchor seeding acts as the interface between quality-source extraction and ACO-inspired synchronization: the archive determines which scheduling patterns are reliable, the ACO-inspired probability model determines how likely each anchor is reused, and the synchronization operator determines how the selected anchor is perturbed and propagated to the population.

Suppose that the k th anchor $\mathcal{A}_{t,k}$ is selected at iteration t , and its quality-layer index is l . A synchronized candidate for the p th solution is generated around the selected anchor as

$$X_p^{sync} = \mathcal{A}_{t,k} + s_\tau(l) \times \tau_t \times \text{randn}(1, dim) + s_a(l) \times \text{drift}_p. \quad (23)$$

where X_p^{sync} is the synchronized candidate, $\mathcal{A}_{t,k}$ is the selected anchor seed, $s_\tau(l)$ controls the layer-dependent local sampling strength, $\text{randn}(1, dim)$ is a Gaussian random vector, $s_a(l)$ controls the global-best drift strength, and drift_p denotes the global guidance direction for the p th solution.

The first term in Equation (23) provides the anchor seed, which preserves the scheduling pattern extracted from a quality-source solution. The second term performs local stochastic sampling around the selected anchor. Its strength is adjusted by the layer-dependent factor $s_\tau(l)$ so that anchors from higher-quality layers can be exploited with a smaller search radius, while anchors from lower-quality layers can maintain a larger exploratory range. The third term introduces a weak drift toward the current global-best region, which provides directional guidance without making the search completely dependent on a single best solution.

The sampling radius is adaptively reduced as

$$\tau_t = 0.1 \times \text{searchRange} \times (1 - \text{ratio}), \quad \text{ratio} = \frac{t}{T}, \quad \text{searchRange} = (UB - LB) \quad (24)$$

where UB and LB are the upper and lower bounds of the continuous search space, and T is the maximum number of iterations. A larger τ_t in the early stage allows broader exploration around multiple anchors, whereas a smaller τ_t in the later stage promotes refined exploitation near promising anchor regions.

After generation, X_p^{sync} is clipped, decoded, repaired if necessary, and evaluated using the fitness function. The synchronized candidate is accepted only if it improves the current solution. Through this anchor-seeding-based synchronization strategy, ASQS reuses high-quality scheduling patterns through ACO-inspired probabilistic selection, while still preserving diversity by allowing different individuals to select different anchors. As a result, the population can exploit multiple promising scheduling regions simultaneously, improving information sharing and reducing dependence on a single global-best solution.

4.7. Overall Procedure of ASQS

This subsection integrates the components introduced in Sections 4.1–4.6 into a unified execution procedure. Given the task set, fog-node set, search-space bounds, objective weights, constraint parameters, and algorithmic control parameters, ASQS aims to obtain the best task-node schedule and its associated performance indicators.

As summarized in Algorithm 1 and Figure 2, ASQS first initializes a population of continuous candidate solutions and maps them into task-node assignments through decoding, feasibility repair, and fitness evaluation. During each iteration, the population is ranked and stratified according to fitness quality, from which quality-source anchors are extracted to update the anchor archive. Guided by the current population state and the anchor archive, ASQS generates candidate schedules through FNO-based directional search, fitness-adaptive Lévy-flight escape, and ACO-inspired multi-anchor synchronization. The generated candidates are subsequently decoded, repaired, evaluated, and compared with the current solutions through greedy selection. The best solution is updated iteratively until the maximum number of iterations is reached.

The computational cost of ASQS is mainly dominated by candidate decoding, feasibility repair, and fitness evaluation. With population size N and maximum iteration number T , the overall complexity can be approximately expressed as $O(T \cdot N \cdot C_{eval})$, where C_{eval} denotes the cost of evaluating one candidate schedule and is primarily determined by the task and fog-node scales.

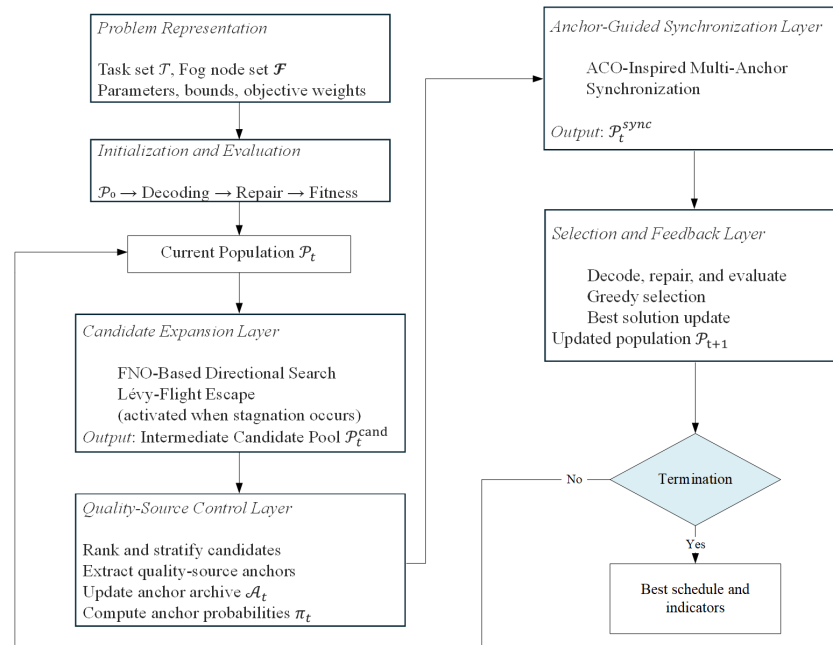


Figure 2. Quality-source-guided closed-loop architecture of ASQS.

Algorithm 1 Overall procedure of ASQS

- 1: **Input:** Task set $\mathcal{T}$, fog node set $\mathcal{F}$, population size N , maximum iterations T , bounds LB and UB , objective weights, constraint parameters, and ASQS control parameters
- 2: **Output:** Best schedule A_{best} , best fitness value fit_{best} , and scheduling indicators
- 3: Initialize population $\mathcal{P}_0 = \{X_p^0\}_{p=1}^N$ within $[LB, UB]$
- 4: Evaluate $\mathcal{P}_0$ by decoding, repairing, and fitness calculation
- 5: Initialize A_{best} , X_{best} , and fit_{best}
- 6: **for** $t = 0$ to $T - 1$ **do**
- 7: Generate FNO-based candidates: $\mathcal{P}_t^{FNO} \leftarrow FNO\text{Search}(\mathcal{P}_t)$
- 8: Evaluate $\mathcal{P}_t^{FNO}$
- 9: Generate Lévy-escape candidates according to E_p^t : $\mathcal{P}_t^{Levy} \leftarrow Levy\text{Escape}(\mathcal{P}_t^{FNO})$
- 10: Evaluate $\mathcal{P}_t^{Levy}$
- 11: Form intermediate candidate pool $\mathcal{P}_t^{cand} \leftarrow \mathcal{P}_t \cup \mathcal{P}_t^{Levy}$
- 12: Construct anchor archive and probabilities: $(\mathcal{A}_t, \pi_t) \leftarrow BuildAnchorArchive(\mathcal{P}_t^{cand})$
- 13: Generate synchronized candidates: $\mathcal{P}_t^{sync} \leftarrow AnchorSync(\mathcal{P}_t, \mathcal{A}_t, \pi_t)$
- 14: Evaluate $\mathcal{P}_t^{sync}$
- 15: Update population: $\mathcal{P}_{t+1} \leftarrow GreedySelect(\mathcal{P}_t, \mathcal{P}_t^{sync})$
- 16: Update A_{best} , X_{best} , and fit_{best}
- 17: **end for**
- 18: Calculate scheduling indicators of A_{best}
- 19: **return** A_{best} , fit_{best} , and scheduling indicators

5. Experiments and Discussion

To comprehensively evaluate the proposed algorithm, a series of experiments are conducted, including benchmark-function optimization, sensitivity analysis, ablation studies, IoT-Fog task scheduling experiments, statistical significance tests, and large-scale

task-intensive scalability analysis. Specifically, 33 benchmark functions are used to test the general optimization ability and convergence stability of ASQS. Sensitivity analysis and ablation studies are conducted to examine parameter robustness and component effectiveness. IoT-Fog scheduling experiments are further designed under node-scaling, task-scaling, and large-scale task-intensive scenarios, including 100 fog nodes and up to 8000 IoT tasks. The algorithms are compared using mean fitness, best fitness, standard deviation, makespan, energy consumption, average latency, deadline satisfaction rate, load balance, runtime, and constraint violation, thereby evaluating scheduling efficiency, QoS satisfaction, workload balance, constraint feasibility, and scalability.

5.1. Experiment I: Global Optimization with a Set of 33 Benchmark Mathematical Functions

5.1.1. Benchmark Functions

The 33 benchmark functions are grouped into four categories with increasing landscape complexity. F01–F08 are unimodal functions for evaluating exploitation and convergence accuracy; F09–F20 are multimodal functions for testing exploration and local-optimum avoidance; F21–F26 are hybrid functions for assessing the balance between exploration and exploitation in coupled landscapes; and F27–F33 are composition functions for examining robustness on highly irregular and deceptive search spaces. Their detailed definitions are provided in Tables 3–6.

Table 3. Unimodal benchmark functions (F01–F08) used to evaluate the exploitation strength of algorithms in smooth landscapes.

Function	Name	Formula	Range	Opt
F01	Shifted Sphere Function	$f_1(x) = \sum_{i=1}^D z_i^2 + f_{\text{bias},1}, \quad z = x - o, \quad X = [x_1, x_2, \dots, x_D],$ $o = [o_1, o_2, \dots, o_D].$	$[-100, 100]^D$	−450
F02	Shifted Schwefel's Problem 1.2	$f_2(x) = \sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2 + f_{\text{bias},2}, \quad z = x - o,$ $X = [x_1, x_2, \dots, x_D], \quad o = [o_1, o_2, \dots, o_D].$	$[-100, 100]^D$	−450
F03	Shifted Rotated High Conditioned Elliptic Function	$f_3(x) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_{\text{bias},3}, \quad z = (x - o)M,$ $X = [x_1, x_2, \dots, x_D], \quad o = [o_1, o_2, \dots, o_D].$	$[-100, 100]^D$	−450
F04	Shifted Schwefel's Problem 1.2 with Noise in Fitness	$f_4(x) = \left[\sum_{i=1}^D \left(\sum_{j=1}^i z_j \right)^2 \right] (1 + 0.4 \mathcal{N}(0, 1)) + f_{\text{bias},4}, \quad z = x - o.$	$[-100, 100]^D$	−450
F05	Schwefel's Problem 2.6 with Global Optimum on Bounds	$f_5(x) = \max\{ A_i X - B_i \} + f_{\text{bias},5}, \quad i = 1, \dots, D,$ $X = [x_1, x_2, \dots, x_D], \text{ where } A \text{ is a } D \times D \text{ matrix.}$	$[-100, 100]^D$	−310
F06	High Conditioned Elliptic Function	$f_6(x) = \sum_{i=1}^D (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_{\text{bias},6}, \quad z = M(x - o).$	$[-100, 100]^D$	100
F07	Bent Cigar Function	$f_7(x) = z_1^2 + \sum_{i=2}^D (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_{\text{bias},7}, \quad z = M(x - o).$	$[-100, 100]^D$	200
F08	Discus Function	$f_8(x) = (10^6)z_1^2 + \sum_{i=2}^D z_i^2 + f_{\text{bias},8}, \quad z = M(x - o).$	$[-100, 100]^D$	300

Table 4. Multimodal benchmark functions (F09–F20) assessing exploration performance and the ability to avoid local optima.

Function	Name	Formula	Range	Opt
F09	Rosenbrock	$F09(x) = \sum_{i=1}^{D-1} [100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2] + f_{\text{bias},9},$ $z = M\left(\frac{2.048(x - o)}{100}\right) + 1, \quad F09 = F09(z).$	$[-100, 100]^D$	400
F10	Ackley	$F10(x) = 20 - 20 \exp\left(-0.2\sqrt{\frac{1}{D} \sum_{i=1}^D z_i^2}\right) - \exp\left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi z_i)\right) + e + f_{\text{bias},10},$ $z = M(x - o).$	$[-100, 100]^D$	500

Table 4. Cont.

Function	Name	Formula	Range	Opt
F11	Weierstrass	$F11(x) = \sum_{i=1}^D \left[\sum_{k=0}^{k_{\max}} a^k \cos(2\pi b^k(z_i + 0.5)) \right] - D \sum_{k=0}^{k_{\max}} a^k \cos(2\pi b^k \cdot 0.5) + f_{\text{bias},11}, a = 0.5,$ $b = 3, k_{\max} = 20, z = M\left(\frac{0.5(x-o)}{100}\right), \quad F11 = F11(z).$	$[-100, 100]^D$	600
F12	Griewank	$F12(x) = \frac{1}{4000} \sum_{i=1}^D z_i^2 - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{\text{bias},12}, \quad z = M\left(\frac{600(x-o)}{100}\right),$ $F12 = F12(z).$	$[-100, 100]^D$	700
F13	Rastrigin	$F13(x) = \sum_{i=1}^D [z_i^2 - 10 \cos(2\pi z_i) + 10] + f_{\text{bias},13}, \quad z = M\left(\frac{5.12(x-o)}{100}\right),$ $F13 = F13(z).$	$[-100, 100]^D$	800
F14	Modified Schwefel	$F14(x) = 418.9829D - \sum_{i=1}^D g(q_i) + f_{\text{bias},14}, \quad q_i = z_i + 420.9687462275036,$ $z = M\left(\frac{1000(x-o)}{100}\right), \quad F14 = F14(z).$	$[-100, 100]^D$	900
F15	Katsuura	$F15(x) = \frac{10}{D^2} \prod_{i=1}^D \left[1 + i \sum_{j=1}^{32} \frac{ 2^j z_i - \text{round}(2^j z_i) }{2^j} \right]^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} + f_{\text{bias},15},$ $z = M\left(\frac{5(x-o)}{100}\right), \quad F15 = F15(z).$	$[-100, 100]^D$	1000
F16	HappyCat	$F16(x) = \left \sum_{i=1}^D z_i^2 - D \right ^{1/4} + \frac{0.5 \sum_{i=1}^D z_i^2 + \sum_{i=1}^D z_i}{D} + 0.5 + f_{\text{bias},16},$ $z = M\left(\frac{5(x-o)}{100}\right), \quad F16 = F16(z).$	$[-100, 100]^D$	1100
F17	HGBat	$F17(x) = \left \left(\sum_{i=1}^D z_i^2 \right)^2 - \left(\sum_{i=1}^D z_i \right)^2 \right ^{1/2} + \frac{0.5 \sum_{i=1}^D z_i^2 + \sum_{i=1}^D z_i}{D} + 0.5 + f_{\text{bias},17},$ $z = M\left(\frac{5(x-o)}{100}\right), \quad F17 = F17(z).$	$[-100, 100]^D$	1200
F18	Expanded Griewank plus Rosenbrock	$F18(x) = F12(f_9(z_1, z_2)) + F12(f_9(z_2, z_3)) + \dots + F12(f_9(z_{D-1}, z_D)) + F12(f_9(z_D, z_1))$ $+ f_{\text{bias},18}, z = M\left(\frac{5(x-o)}{100}\right) + 1, \quad F18 = F18(z).$	$[-100, 100]^D$	1300
F19	Expanded Scaffer F6	$F19(x) = g(z_1, z_2) + g(z_2, z_3) + \dots + g(z_{D-1}, z_D) + g(z_D, z_1) + f_{\text{bias},19},$ $g(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2 + y^2}) - 0.5}{[1 + 0.001(x^2 + y^2)]^2}, \quad z = M(x - o) + 1, \quad F19 = F19(z).$	$[-100, 100]^D$	1400
F20	D-type	$F20(x) = \begin{cases} C_i(x), & x \in S_i, i = 1, 2, \dots, m, \\ g(x), & x \in S_2 \cup \dots \cup S_m, \end{cases}$ $C_i(x) = \left(\frac{1}{\beta_i} \ x - M_i\ ^2 - \alpha_i \right) \ x - M_i\ ^2 + f_i, \quad g(x) = \ x - T\ ^2 + t, \quad m = 20.$	$[-100, 100]^D$	1500

Table 5. Hybrid benchmark functions (F21–F26) combining multiple subfunctions to test exploration and exploitation performance.

Function	Formulas	Range	Opt
F21 Hybrid Function (HF01)	$N = 3, F14, F13, F01, p = [0.3, 0.3, 0.4]$	$[-100, 100]^D$	1700
F22 Hybrid Function (HF02)	$N = 3, f07, F17, F13, p = [0.3, 0.3, 0.4]$	$[-100, 100]^D$	1800
F23 Hybrid Function (HF03)	$N = 4, F12, F11, f09, F19, p = [0.2, 0.2, 0.3, 0.3]$	$[-100, 100]^D$	1900
F24 Hybrid Function (HF04)	$N = 4, F17, f08, F18, F13, p = [0.2, 0.2, 0.3, 0.3]$	$[-100, 100]^D$	2000
F25 Hybrid Function (HF05)	$N = 5, F19, F17, f09, F14, f06, p = [0.1, 0.2, 0.2, 0.2, 0.3]$	$[-100, 100]^D$	2100
F26 Hybrid Function (HF06)	$N = 5, F15, F16, F18, F14, F10, p = [0.1, 0.2, 0.2, 0.2, 0.3]$	$[-100, 100]^D$	2200

Table 6. Composition benchmark functions (F27–F33) incorporating rotated and shifted components.

Function	Parameters	Range	Opt
F27 Composition Function (CF01)	$N = 5, F09, F06, F07, F08, F06, \sigma = [10, 20, 30, 40, 50]$ $\lambda = [1, 1e-6, 1e-26, 1e-6, 1e-6]$ $bias = [0, 100, 200, 300, 400]$	$[-100, 100]^D$	2300
F28 Composition Function (CF02)	$N = 3, F10, F09, F11, \sigma = [20, 20, 20]$ $\lambda = [1, 1, 1]$ $bias = [0, 100, 200]$	$[-100, 100]^D$	2400
F29 Composition Function (CF03)	$N = 3, F11, F09, F01, \sigma = [10, 30, 50]$ $\lambda = [0.25, 1, 1e-7]$ $bias = [0, 100, 200]$	$[-100, 100]^D$	2500
F30 Composition Function (CF04)	$N = 5, F11, F13, F01, F06, F07, \sigma = [10, 10, 10, 10, 10]$ $\lambda = [0.25, 1, 1e-7, 2.5, 10]$ $bias = [0, 100, 200, 300, 400]$	$[-100, 100]^D$	2600
F31 Composition Function (CF05)	$N = 5, F14, F09, F11, F06, F01, \sigma = [10, 10, 10, 10, 10]$ $\lambda = [10, 10, 2.5, 25, 1e-6]$ $bias = [0, 100, 200, 300, 400]$	$[-100, 100]^D$	2700
F32 Composition Function (CF06)	$N = 5, F15, F13, F11, F16, F01, \sigma = [10, 20, 30, 40, 50]$ $\lambda = [2.5, 10, 2.5, 5e-4, 1e-6]$ $bias = [0, 100, 200, 300, 400]$	$[-100, 100]^D$	2800
F33 Composition Function (CF07)	$N = 3, F17, F18, F19, \sigma = [10, 30, 50]$ $\lambda = [1, 1, 1]$ $bias = [0, 100, 200]$	$[-100, 100]^D$	2900

5.1.2. Performance Evaluation Metrics

To evaluate the optimization accuracy, stability, and statistical significance of the compared algorithms, the experimental results are analyzed using the mean value (Avg), standard deviation (Std), ranking, Wilcoxon signed-rank test, and effect size. For M independent runs, Avg and Std are calculated as follows:

$$\text{Avg} = \frac{1}{M} \sum_{i=1}^M f_i, \quad (25)$$

$$\text{Std} = \sqrt{\frac{1}{M-1} \sum_{i=1}^M (f_i - \text{Avg})^2}, \quad (26)$$

where f_i is the fitness value obtained in the i th run. Algorithms are ranked according to their average fitness values, and lower values indicate better performance for minimization problems. The Wilcoxon signed-rank test is conducted at a significance level of $\alpha = 0.05$ to assess whether the performance differences are statistically significant. In addition, Better/Equal/Worse counts and effect sizes are reported to summarize pairwise comparison results and quantify the magnitude of performance differences.

5.1.3. Comparison with Other Algorithms and Parameter Setting

Table 7 summarizes the parameter settings used in the benchmark-function experiments, and Table 8 presents the comparative results. The selected baselines include classical metaheuristics, such as FNO [32], ACO [33], HSA [34], DE [35], GWO [36], and WOA [37]; advanced evolutionary and hybrid optimizers, including MetaDE [38], HPSOBOA [39], PS-BES [40], and PSOBOA [41]; and the composite metaheuristic algorithm CMA. These competitors represent diverse optimization mechanisms rather than a narrow set of similar algorithms, making the comparison more stringent.

To avoid biased conclusions caused by unequal configurations, all algorithms were tested under the same dimension, population size, number of runs, and computational budget. Baseline parameters follow the original literature or standard settings, while ASQS uses one fixed configuration throughout the benchmark experiments. No algorithm was given additional iterations or a larger population. Thus, the reported differences mainly reflect the effectiveness of the search mechanisms rather than differences in experimental resources.

Table 7. Setting of algorithm parameters.

Algorithms	Parameters
All	$dim = 30, nPop = 30, numRuns = 30$
ASQS	$numPopulations = 3, w = 0.9 - 0.5 \times ratio, c_1 = c_2 = c_3 = 1.5,$ $\beta = 1.5, \lambda = 0.01, \tau = 0.1 \times searchRange \times (1 - ratio),$ $s_\tau = [0.6, 1.0, 1.4], s_a = [0.2, 0.3, 0.4], V_0 = 0$
CMA	$numPopulations = 3, V_0 = 0,$ $V_{max} = 1, w_{max} = 0.9, w_{min} = 0.2, C_1 = 2, C_2 = 2, \alpha_{CMA} = 2,$ $p = 0.6, numAnts = 50, \alpha_{ACO} = 1, \beta_{ACO} = 2, \rho = 0.1, Q = 1$
ACO	$q = 0.5, \chi_i = 0.8$
FNO	$\alpha = \sqrt{1 - \frac{FEs}{MaxFEs}}, \phi \in \{0, 1\}, r_1 \in [0, 1], \mu \in [0, 1],$ $DFF = \mu - \alpha + 2\alpha \cdot rand , W = (2 + U(-1, 1)) \times U(1, 2)$
MetaDE	$F = 0.5, CR = 0.9$
HSA	$HMCR = 0.995, PAR = 0.9, bw = 0.01$
DE	$F = 0.5, CR = 0.9$
GWO	$a = linear-decrease-from-2-to-0, C = random[0, 2]$
HPSBOA	$w_{max} = 0.9, w_{min} = 0.4, c_1 = 2.0, c_2 = 0.2, vMaxRatio = 0.1$ $a = 0.1, c_0 = 0.01, pBOA_{max} = 0.8, pBOA_{min} = 0.2$
PS-BES	$w_{max} = 0.9, w_{min} = 0.4, c_1 = 2.0, c_2 = 2.0, vMaxRatio = 0.1,$ $a = 10, R = 1.5, \beta_{max} = 0.9, \beta_{min} = 0.1, K = 20, q = 0.2$
PSOBOA	$w_{max} = 0.9, w_{min} = 0.4, c_{1max} = 2.5, c_{1min} = 0.5$ $c_{2max} = 2.5, c_{2min} = 0.5, vMaxRatio = 0.1, a = 0.1, c_0 = 0.01$ $pBOA_{max} = 0.8, pBOA_{min} = 0.2, \beta_{max} = 0.9, \beta_{min} = 0.1$
WOA	$b = 1$

5.1.4. Data Analysis

Table 8 compares ASQS with 12 competing algorithms on 33 benchmark functions. ASQS obtains the best Avg_Rank of 3.4 and ranks first overall, demonstrating the strongest aggregate performance among all methods. It achieves the best results on 10 functions, namely F3, F9, F11, F12, F17, F20, F21, F30, F31, and F32, and remains competitive on most of the remaining functions. This confirms that the advantage of ASQS is not restricted to a small subset of test cases.

The superiority of ASQS is especially evident on hybrid and composition functions, such as F21, F30, F31, and F32, where the search landscapes are more complex and deceptive. The result on F32 is particularly representative: ASQS obtains the best average value with zero standard deviation, indicating both high accuracy and strong stability. These results support the effectiveness of the quality-source archive and multi-anchor synchronized search in maintaining reliable guidance under complex optimization conditions.

ASQS does not dominate every function. ACO performs better on F14, F15, and F28; DE shows stronger results on F18, F19, F22, F23, F26, and F27; and PS-BES achieves the best ranks on F7, F8, F24, F25, F29, and F33. This indicates that some baseline algorithms remain advantageous on specific landscape structures. However, ASQS delivers the best overall ranking, showing a more balanced and robust performance across the entire benchmark suite.

Table 8. Comparison results of all algorithms on 33 benchmark functions.

Func.	Metrics	ASQS	CMA	FNO	PSO	ACO	HSA	GWO	WOA	DE	MetaDE	HPSOBOA	PS-BES	PSOBOA
F1	Avg	0	1.2039×10^{-1}	0	3.3333×10^2	0	0	2.3701×10^{-2}	0	4.0027	0	0	2.3117×10^{-6}	0
	Std	0	3.7468×10^{-2}	0	1.8257×10^3	0	0	6.0355×10^{-3}	0	1.3815×10^1	0	0	1.0979×10^{-5}	0
	Rank	4.50	11	4.50	13	4.50	4.50	10	4.50	12	4.50	4.50	9	4.50
F2	Avg	0	0	0	5.168×10^4	0	1.7253×10^5	5.0968×10^3	5.5282×10^2	0	0	2.781×10^2	0	0
	Std	7.3379×10^{-1}	5.0289	1.4165×10^3	1.9279×10^4	2.1992×10^{-6}	6.1266×10^4	2.3742×10^3	9.3086×10^2	1.0717×10^2	2.7527×10^2	5.9221×10^2	1.1146×10^{-1}	6.0288×10^2
	Rank	4.50	4.50	4.50	12	4.50	13	11	10	4.50	4.50	9	4.50	4.50
F3	Avg	6.0123×10^5	4.1485×10^6	3.5658×10^6	1.7317×10^8	2.0531×10^6	1.5456×10^9	1.1892×10^7	7.602×10^6	2.1604×10^6	3.4903×10^6	6.0806×10^6	1.1707×10^6	7.1385×10^6
	Std	2.6996×10^5	1.5974×10^6	2.1901×10^6	1.0517×10^8	1.4705×10^6	4.4747×10^8	7.9495×10^6	4.0255×10^6	7.7296×10^5	1.5455×10^6	4.293×10^6	6.2524×10^5	4.5718×10^6
	Rank	1	7	6	12	3	13	11	10	4	5	8	2	9
F4	Avg	0	0	4.5244×10^3	5.3418×10^4	0	1.377×10^5	1.338×10^4	7.0718×10^3	0	1.1237×10^4	7.9076×10^3	0	1.0378×10^4
	Std	8.5930	2.2691×10^2	5.1118×10^3	2.5874×10^4	1.6723×10^2	2.5766×10^4	3.8256×10^3	3.9819×10^3	3.6307×10^1	4.2904×10^3	2.9408×10^3	2.5974×10^1	4.3495×10^3
	Rank	3	3	6	12	3	13	11	7	3	10	8	3	9
F5	Avg	0	0	3.013×10^3	2.5506×10^4	0	6.0562×10^4	1.3024×10^4	2.5033×10^3	0	2.682×10^4	5.1778×10^3	3.1605×10^1	3.6956×10^3
	Std	0	5.1506×10^1	6.5065×10^3	7.5231×10^3	3.9354×10^2	5.3321×10^3	2.5758×10^3	4.6823×10^3	2.1002×10^2	3.3409×10^3	7.738×10^3	1.871×10^3	6.6426×10^3
	Rank	2.50	2.50	7	11	2.50	13	10	6	2.50	12	9	5	8
F6	Avg	1.9999×10^6	3.5645×10^6	2.4351×10^6	1.5322×10^8	1.3775×10^6	1.3597×10^9	1.1873×10^7	6.7835×10^6	3.9723×10^5	2.8243×10^6	4.7227×10^6	1.061×10^6	6.9227×10^6
	Std	8.308×10^5	1.3346×10^6	1.6634×10^6	1.0738×10^8	5.8209×10^5	1.7972×10^8	6.083×10^6	2.9872×10^6	2.3962×10^5	1.5815×10^6	2.0751×10^6	7.8865×10^5	3.6604×10^6
	Rank	4	7	5	12	3	13	11	9	1	6	8	2	10
F7	Avg	5.9358×10^4	1.7202×10^5	1.5835×10^8	1.6433×10^{10}	9.855×10^3	4.8267×10^{10}	4.8139×10^8	1.888×10^7	6.381×10^6	3.2582×10^7	5.4276×10^6	8.0163×10^3	1.8155×10^5
	Std	1.5863×10^5	4.2027×10^5	5.5176×10^8	6.6448×10^9	9.2354×10^3	4.7739×10^9	4.575×10^8	4.2632×10^7	3.3836×10^7	1.2284×10^8	2.2521×10^7	6.1161×10^3	6.7852×10^5
	Rank	3	4	10	12	2	13	11	8	7	9	6	1	5
F8	Avg	5.497×10^2	1.191×10^3	2.0798×10^4	8.1726×10^4	6.8999×10^3	6.5266×10^4	2.57×10^4	7.2127×10^3	1.4675×10^3	1.7439×10^4	5.3129×10^3	3.447×10^2	5.4466×10^3
	Std	9.4811×10^1	3.6252×10^2	5.5662×10^3	3.7877×10^4	2.6462×10^3	7.2796×10^3	6.4067×10^3	3.3406×10^3	9.9065×10^2	5.1864×10^3	1.3321×10^3	2.0569×10^1	1.8068×10^3
	Rank	2	3	10	13	7	12	11	8	4	9	5	1	6
F9	Avg	4.29×10^2	4.6892×10^2	4.9658×10^2	2.8682×10^3	4.6147×10^2	1.7997×10^4	6.4439×10^2	5.0343×10^2	4.6175×10^2	4.907×10^2	4.8847×10^2	4.6538×10^2	4.4735×10^2
	Std	3.295×10^1	1.9442×10^1	5.9847×10^1	1.1939×10^3	3.0625×10^1	2.9844×10^3	7.7666×10^1	3.7986×10^1	4.7303×10^1	5.4244×10^1	3.1503×10^1	2.3803×10^1	3.7811×10^1
	Rank	1	6	9	12	3	13	11	10	4	8	7	5	2
F10	Avg	5.2095×10^2	5.2026×10^2	5.2097×10^2	5.2088×10^2	5.2014×10^2	5.2101×10^2	5.2094×10^2	5.2026×10^2	5.2096×10^2	5.2003×10^2	5.2095×10^2	5.2093×10^2	5.2093×10^2
	Std	4.3691×10^{-2}	9.0695×10^{-2}	6.046×10^{-2}	1.0309×10^{-1}	9.938×10^{-2}	7.4294×10^{-2}	5.5456×10^{-2}	1.1776×10^{-1}	3.6591×10^{-2}	6.4637×10^{-2}	5.0587×10^{-2}	5.6894×10^{-2}	4.7343×10^{-2}
	Rank	9	3	12	5	2	13	8	4	11	1	10	6	7
F11	Avg	6.0815×10^2	6.1128×10^2	6.2312×10^2	6.2946×10^2	6.0992×10^2	6.4305×10^2	6.1383×10^2	6.2615×10^2	6.1027×10^2	6.3464×10^2	6.2912×10^2	6.1342×10^2	6.3056×10^2
	Std	2.3135	3.1567	4.9851	3.1500	4.1745	1.1983	2.6097	3.0299	3.3578	2.9218	3.2554	4.0636	3.9144
	Rank	1	4	7	10	2	13	6	8	3	12	9	5	11
F12	Avg	7×10^2	7.003×10^2	7.0705×10^2	9.112×10^2	7.0009×10^2	1.2404×10^3	7.1597×10^2	7.0114×10^2	7.0066×10^2	7.0001×10^2	7.0001×10^2	7.0001×10^2	7.0001×10^2
	Std	6.1646×10^{-3}	7.9615×10^{-2}	1.012×10^1	7.3081×10^1	3.7641×10^{-2}	4.5547×10^1	1.7796×10^1	2.7353×10^{-1}	2.1607	2.106×10^{-2}	1.1604×10^{-2}	1.0967×10^{-2}	1.7134×10^{-2}
	Rank	1	7	10	12	6	13	11	9	8	4	3	2	5

Table 8. Cont.

Func.	Metrics	ASQS	CMA	FNO	PSO	ACO	HSA	GWO	WOA	DE	MetaDE	HPSOBOA	PS-BES	PSOBOA
F13	Avg	9.716×10^2	8.7919×10^2	8.7589×10^2	1.0078×10^3	8.5097×10^2	1.1587×10^3	9.1129×10^2	9.5922×10^2	8.3806×10^2	9.7584×10^2	9.7108×10^2	9.0099×10^2	9.5826×10^2
	Std	1.02×10^1	1.9561×10^1	2.7833×10^1	4.7602×10^1	1.302×10^1	1.9977×10^1	5.029×10^1	3.6951×10^1	1.8907×10^1	3.0226×10^1	5.834×10^1	2.6625×10^1	3.3175×10^1
	Rank	10	4	3	12	2	13	6	8	1	11	9	5	7
F14	Avg	7.34×10^3	4.0112×10^3	7.5278×10^3	6.1461×10^3	3.5217×10^3	9.4984×10^3	4.1743×10^3	5.0993×10^3	6.6929×10^3	5.557×10^3	7.1113×10^3	4.0558×10^3	5.9973×10^3
	Std	2.7351×10^2	6.7415×10^2	7.1767×10^2	6.9473×10^2	6.3139×10^2	4.299×10^2	1.1708×10^3	7.4809×10^2	1.2339×10^3	6.5148×10^2	4.0515×10^2	6.6757×10^2	7.9618×10^2
	Rank	11	2	12	8	1	13	4	5	9	6	10	3	7
F15	Avg	1.0025×10^3	1.0005×10^3	1.0027×10^3	1.0016×10^3	1.0002×10^3	1.0035×10^3	1.0025×10^3	1.0008×10^3	1.0024×10^3	1.0017×10^3	1.0024×10^3	1.0017×10^3	1.0022×10^3
	Std	2.4656×10^{-1}	3.4331×10^{-1}	2.8183×10^{-1}	4.589×10^{-1}	8.1847×10^{-2}	5.1174×10^{-1}	2.3915×10^{-1}	3.3747×10^{-1}	3.1214×10^{-1}	5.2577×10^{-1}	3.4201×10^{-1}	7.4085×10^{-1}	3.3492×10^{-1}
	Rank	10	2	12	4	1	13	11	3	8	5	9	6	7
F16	Avg	1.1003×10^3	1.1005×10^3	1.1005×10^3	1.1034×10^3	1.1003×10^3	1.1054×10^3	1.1004×10^3	1.1005×10^3	1.1003×10^3	1.1008×10^3	1.1008×10^3	1.1003×10^3	1.1007×10^3
	Std	5.6853×10^{-2}	1.3206×10^{-1}	1.1585×10^{-1}	6.0123×10^{-1}	6.0565×10^{-2}	2.4728×10^{-1}	8.0801×10^{-2}	1.0129×10^{-1}	5.724×10^{-2}	1.6566×10^{-1}	1.6275×10^{-1}	7.3257×10^{-2}	1.4373×10^{-1}
	Rank	2	7	6	12	3	13	5	8	1	11	10	4	9
F17	Avg	1.2004×10^3	1.2005×10^3	1.2006×10^3	1.2468×10^3	1.2004×10^3	1.3202×10^3	1.2009×10^3	1.2005×10^3	1.2004×10^3	1.2009×10^3	1.2008×10^3	1.2005×10^3	1.2009×10^3
	Std	1.7065×10^{-1}	3.5845×10^{-1}	2.3635×10^{-1}	1.4293×10^1	2.7026×10^{-1}	8.8024	1.9019	2.838×10^{-1}	1.6754×10^{-1}	3.7228×10^{-1}	4.1125×10^{-1}	2.4011×10^{-1}	3.5008×10^{-1}
	Rank	1	6	7	12	2	13	11	4	3	10	8	5	9
F18	Avg	1.315×10^3	1.3099×10^3	1.3179×10^3	6.7988×10^4	1.3056×10^3	1.5269×10^6	1.3427×10^3	1.3125×10^3	1.3039×10^3	2.65×10^3	1.3331×10^3	1.3067×10^3	1.3359×10^3
	Std	9.7241×10^{-1}	2.3843	1.5138×10^1	1.0206×10^5	1.8242	7.595×10^5	4.2162×10^1	3.2944	2.3394	1.1425×10^3	1.215×10^1	3.6503	1.4277×10^1
	Rank	6	4	7	12	2	13	10	5	1	11	8	3	9
F19	Avg	1.4121×10^3	1.4112×10^3	1.4124×10^3	1.4117×10^3	1.411×10^3	1.4132×10^3	1.4109×10^3	1.4119×10^3	1.4116×10^3	1.4125×10^3	1.4115×10^3	1.411×10^3	1.412×10^3
	Std	3.5568×10^{-1}	5.8404×10^{-1}	5.1671×10^{-1}	4.7251×10^{-1}	6.7601×10^{-1}	2.5503×10^{-1}	6.9625×10^{-1}	5.4209×10^{-1}	4.7753×10^{-1}	5.8201×10^{-1}	3.7422×10^{-1}	5.9026×10^{-1}	4.6359×10^{-1}
	Rank	10	4	11	7	2	13	1	8	6	12	5	3	9
F20	Avg	1.51×10^3	1.5101×10^3	1.5982×10^3	2.181×10^4	1.51×10^3	4.888×10^4	2.5483×10^3	1.5207×10^3	1.5443×10^3	1.51×10^3	1.51×10^3	1.51×10^3	1.51×10^3
	Std	0	3.3133×10^{-2}	3.3552×10^2	6.925×10^3	4.9286×10^{-3}	3.3337×10^3	9.698×10^2	2.1482×10^1	9.8496×10^1	0	0	4.008×10^{-6}	0
	Rank	1	7	10	12	6	13	11	8	9	3	4	5	2
F21	Avg	2.0047×10^3	2.5735×10^3	2.6329×10^3	6.0958×10^3	2.3942×10^3	1.7163×10^4	2.6549×10^3	2.9779×10^3	2.1362×10^3	3.1279×10^3	2.8424×10^3	2.4654×10^3	2.9231×10^3
	Std	2.8499×10^2	2.8023×10^2	2.9101×10^2	1.6423×10^3	2.2014×10^2	2.1706×10^3	2.2677×10^2	3.5691×10^2	2.2472×10^2	5.1797×10^2	3.1638×10^2	2.5886×10^2	3.463×10^2
	Rank	1	5	6	12	3	13	7	10	2	11	8	4	9
F22	Avg	8.8627×10^3	1.1051×10^4	1.4295×10^4	3.6477×10^8	1.1446×10^4	5.4531×10^9	5.3782×10^6	1.3554×10^4	5.9887×10^3	1.3809×10^4	1.1365×10^4	1.1152×10^4	1.099×10^4
	Std	6.112×10^3	7.6653×10^3	7.967×10^3	4.6546×10^8	8.2488×10^3	1.7496×10^9	1.4012×10^7	7.7751×10^3	4.6955×10^3	7.5042×10^3	7.6607×10^3	7.0904×10^3	7.0068×10^3
	Rank	2	4	10	12	7	13	11	8	1	9	6	5	3
F23	Avg	1.9123×10^3	1.9143×10^3	1.9448×10^3	2.1313×10^3	1.9088×10^3	4.8009×10^3	1.92×10^3	1.9167×10^3	1.9082×10^3	1.9479×10^3	1.9131×10^3	1.9169×10^3	1.9159×10^3
	Std	1.0265×10^1	3.6519	2.52×10^1	1.3584×10^2	2.8043	1.032×10^3	1.1784×10^1	9.7608	3.9897	2.2983×10^1	2.6099	1.1348×10^1	3.7461
	Rank	3	5	10	12	2	13	9	7	1	11	4	8	6
F24	Avg	2.2558×10^3	2.3186×10^3	6.6671×10^3	3.8518×10^4	2.2475×10^3	7.8681×10^5	9.1881×10^3	2.4399×10^3	3.7794×10^3	5.7776×10^3	2.607×10^3	2.1295×10^3	2.7177×10^3
	Std	9.7911×10^1	5.9007×10^1	5.6156×10^3	3.0258×10^4	5.4503×10^1	9.7339×10^5	4.3128×10^3	4.0069×10^2	2.4166×10^3	2.3296×10^3	5.1817×10^2	3.7408×10^1	4.9144×10^2
	Rank	3	4	10	12	2	13	11	5	8	9	6	1	7

Table 8. Cont.

Func.	Metrics	ASQS	CMA	FNO	PSO	ACO	HSA	GWO	WOA	DE	MetaDE	HPSOBOA	PS-BES	PSOBOA
F25	Avg	2.7979×10^4	1.4143×10^5	4.5142×10^5	1.8721×10^6	4.3482×10^4	5.9983×10^7	5.1571×10^5	2.5782×10^5	3.0077×10^4	1.0017×10^5	1.4365×10^5	2.0503×10^4	1.474×10^5
	Std	2.0526×10^4	1.0239×10^5	1.2316×10^6	2.2516×10^6	2.5696×10^4	6.0468×10^7	6.4685×10^5	2.454×10^5	2.6471×10^4	1.1999×10^5	9.6712×10^4	1.3767×10^4	1.1376×10^5
	Rank	2	6	10	12	4	13	11	9	3	5	7	1	8
F26	Avg	2.3873×10^3	2.501×10^3	2.467×10^3	2.9798×10^3	2.4717×10^3	3.3345×10^4	2.505×10^3	2.591×10^3	2.3795×10^3	3.0678×10^3	2.5918×10^3	2.4683×10^3	2.7321×10^3
	Std	1.8723×10^2	1.4894×10^2	1.394×10^2	4.036×10^2	1.4517×10^2	3.537×10^4	1.3879×10^2	1.7758×10^2	1.5544×10^2	3.0451×10^2	1.8375×10^2	1.2693×10^2	2.1606×10^2
	Rank	2	6	3	11	5	13	7	8	1	12	9	4	10
F27	Avg	9.1201×10^6	2.945×10^7	1.1427×10^7	3.1931×10^8	2.3146×10^7	9.901×10^9	5.0636×10^7	2.6758×10^7	7.0888×10^6	3.3762×10^7	8.4085×10^7	2.1434×10^7	4.9171×10^7
	Std	9.0034×10^6	2.3591×10^7	1.2121×10^7	3.1849×10^8	1.8906×10^7	3.5908×10^9	5.3747×10^7	2.9469×10^7	1.2699×10^6	3.7577×10^7	7.1404×10^7	1.9748×10^7	3.36×10^7
	Rank	2	7	3	12	5	13	10	6	1	8	11	4	9
F28	Avg	2.5445×10^3	2.5433×10^3	2.5603×10^3	2.8122×10^3	2.5411×10^3	4.031×10^3	2.5582×10^3	2.5499×10^3	2.5484×10^3	2.5956×10^3	2.5542×10^3	2.5441×10^3	2.5566×10^3
	Std	2.3690	2.0734	1.4929×10^1	1.124×10^2	1.3056	3.85×10^2	1.895×10^1	3.8567	3.4164	4.8124×10^1	5.7675	2.3990	7.3250
	Rank	4	2	10	12	1	13	9	6	5	11	7	3	8
F29	Avg	4.4778×10^3	4.357×10^3	4.7083×10^3	1.4095×10^4	4.3114×10^3	3.9789×10^5	5.5609×10^3	4.4212×10^3	4.4792×10^3	6.2327×10^3	4.9548×10^3	3.3578×10^3	4.8355×10^3
	Std	5.0687	5.6036×10^2	6.4664×10^2	8.7612×10^3	5.4489×10^2	1.0914×10^5	1.7331×10^3	3.233×10^2	9.8259	1.8347×10^3	3.0359×10^3	8.714×10^2	2.5843×10^3
	Rank	5	3	7	12	2	13	10	4	6	11	9	1	8
F30	Avg	2.628×10^3	2.7655×10^6	1.8797×10^6	8.9656×10^6	6.6606×10^5	4.1932×10^{11}	1.8077×10^6	1.2965×10^5	2.6331×10^3	3.0582×10^6	1.2073×10^7	2.111×10^6	7.3381×10^6
	Std	9.6132×10^1	6.8636×10^6	1.0281×10^7	3.4464×10^7	3.6329×10^6	7.7762×10^{11}	6.962×10^6	6.9558×10^5	1.0066×10^2	6.8961×10^6	1.2539×10^7	6.0494×10^6	9.4444×10^6
	Rank	1	8	6	11	4	13	5	3	2	9	12	7	10
F31	Avg	2.877×10^3	3.9255×10^3	3.5971×10^3	1.1694×10^5	5.0365×10^3	2.328×10^{12}	4.5653×10^3	4.3982×10^3	3.5929×10^3	6.6224×10^3	2.125×10^4	6.4155×10^3	9.6264×10^3
	Std	1.1341×10^1	4.8587×10^3	3.7907×10^3	5.9042×10^5	5.6009×10^3	5.0142×10^{12}	9.1833×10^3	6.3249×10^3	3.0883×10^3	1.0849×10^4	7.4479×10^4	8.914×10^3	1.5118×10^4
	Rank	1	4	3	12	7	13	6	5	2	9	11	8	10
F32	Avg	3.8039×10^6	7.0678×10^{10}	5.1254×10^{13}	6.8821×10^{15}	1.4468×10^{10}	1.876×10^{16}	1.0729×10^{15}	1.13×10^{13}	7.8908×10^{11}	3.8039×10^6	3.8069×10^6	3.9691×10^6	3.8039×10^6
	Std	0	1.7352×10^{10}	1.9505×10^{14}	2.7408×10^{15}	3.0843×10^9	1.5401×10^{15}	1.0684×10^{15}	1.7564×10^{13}	1.6777×10^{12}	8.4747×10^1	1.0086×10^4	8.0856×10^5	1.8937×10^{-8}
	Rank	1	7	10	12	6	13	11	9	8	3	4	5	2
F33	Avg	3.0391×10^3	3.0354×10^3	1.6102×10^4	3.2862×10^5	2.9026×10^4	2.4247×10^6	3.2699×10^3	1.8803×10^4	3.0352×10^3	4.3063×10^3	3.058×10^3	3.0326×10^3	3.0585×10^3
	Std	1.8341	2.2194	7.1547×10^4	4.6107×10^5	9.8926×10^4	1.187×10^6	4.9725×10^2	8.6343×10^4	1.5092×10^1	1.1298×10^3	8.9223	2.7690	1.1199×10^1
	Rank	4	3	9	12	11	13	7	10	2	8	5	1	6
Paired rank	–	–	21/2/10	30/1/2	29/0/4	16/4/13	32/0/1	28/0/5	26/1/6	18/4/11	28/2/3	30/1/2	17/2/14	26/2/5
Avg_Rank	–	3.4	4.5	7.5	10.5	3.5	12.7	8.6	6.8	4.3	7.83	7.8	3.8	6.9
Overall_rank	–	1	5	8	12	2	13	11	6	4	10	9	3	7

5.1.5. Significance Analysis

According to the Wilcoxon signed-rank test results presented in Table 9, the performance differences between ASQS and each competing algorithm were evaluated on 33 benchmark functions at a significance level of 0.05. The terms Better, Equal, and Worse denote the numbers of functions on which ASQS performs better than, equal to, or worse than the compared algorithm, respectively. The p -value is used to measure statistical significance, while the Symbol column indicates the final comparison result.

As shown in Table 9, ASQS obtains more Better results than most competing algorithms. Compared with HSA, FNO, PSO, GWO, and MetaDE, the Better/Equal/Worse results are 32/0/1, 30/1/2, 29/0/4, 28/0/5, and 28/2/3, respectively. ASQS also performs competitively against WOA, HPSOBOA, and PE-BES. Compared with CMA and DE, ASQS still shows advantages, although the margins are relatively smaller. For ACO and PSOBOA, the results are comparatively close, indicating comparable performance between these algorithms and ASQS on some benchmark functions.

All p -values are below 0.05, indicating that the differences between ASQS and all competing algorithms are statistically significant. The effect sizes are high when ASQS is compared with PSO and HSA, and are also notable against GWO, WOA, CMA, ACO, FNO, DE, and PE-BES. For MetaDE, HPSOBOA, and PSOBOA, the effect sizes are moderate, indicating that ASQS still retains an advantage, although the improvement is relatively limited.

The results in Table 9 confirm that ASQS is statistically superior to most competing algorithms and demonstrates strong practical performance across the 33 benchmark functions.

Table 9. Wilcoxon signed-rank test results between ASQS and competing algorithms.

Algorithm	Better	Equal	Worse	p -Value	Symbol	Effect Size
ASQS-CMA	21	2	10	1.5497×10^{-6}	+	0.92000
ASQS-FNO	30	1	2	1.0490×10^{-5}	+	0.84615
ASQS-PSO	29	0	4	4.6566×10^{-10}	+	1.00000
ASQS-ACO	16	4	13	5.7220×10^{-6}	=	0.91304
ASQS-HSA	32	0	1	4.6566×10^{-10}	+	1.00000
ASQS-GWO	28	0	5	1.1176×10^{-7}	+	0.93103
ASQS-WOA	26	1	6	8.0466×10^{-7}	+	0.92308
ASQS-DE	18	4	11	1.2112×10^{-4}	+	0.81818
ASQS-MetaDE	28	2	3	9.1223×10^{-4}	+	0.64286
ASQS-HPSOBOA	30	1	2	1.0372×10^{-4}	+	0.72414
ASQS-PSOBOA	17	2	14	5.9476×10^{-5}	=	0.73333
ASQS-PS-BES	26	2	5	2.2125×10^{-4}	+	0.80952

5.1.6. Characteristic Analysis

Figure 3 presents the convergence curves of ASQS and the competing algorithms on nine representative benchmark functions. Overall, ASQS shows competitive convergence behavior and reaches lower final fitness values on most displayed cases, which is consistent with the ranking results in Table 8. In particular, ASQS exhibits rapid descent on F05, F09, F12, F14, and F20, while several baseline algorithms converge more slowly or stagnate at higher fitness values. This indicates that ASQS can maintain effective search progress across different benchmark landscapes.

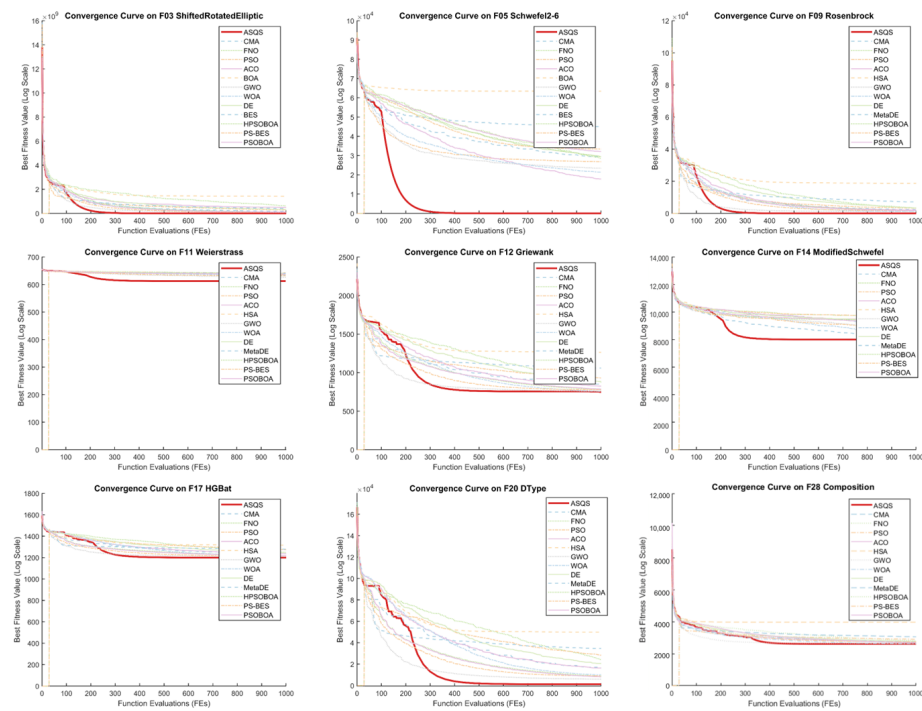


Figure 3. Representative convergence curves of ASQS and competing algorithms on selected benchmark functions.

The observed convergence pattern is mainly attributed to the coordinated search structure of ASQS. The quality-layer organization maintains candidate solutions from different fitness levels, the FNO-based update provides directional search guidance, and the Lévy-flight perturbation helps escape stagnant regions. In addition, the quality-source archive and multi-anchor synchronized search allow the population to exploit multiple promising regions rather than relying on a single best solution. Consequently, ASQS achieves a stable balance between exploration and exploitation, leading to competitive convergence performance.

5.1.7. Sensitivity Analysis

To justify the default parameter configuration of ASQS, we conducted a one-factor-at-a-time sensitivity analysis on four key parameters: the Lévy scale λ , the synchronization radius τ_0 , the layer scale s_τ , and the attraction scale α . Three representative benchmark functions, F03, F12, and F30, were used to cover unimodal, multimodal, and composition landscapes.

Figure 4 and the rank results show a clear pattern: the default setting achieves the lowest normalized average fitness for every parameter and ranks first on all three benchmark functions. Specifically, $\lambda = 0.010$, $\tau_0 = 0.10$, the default-layered s_τ , and the default α all obtain an average rank of 1.00. This demonstrates that the adopted parameter configuration is the best-performing choice among the tested alternatives.

The performance degradation away from the default settings is also consistent. Too small λ weakens local-optimum escape, while too large λ causes unstable jumps. Too small τ_0 limits anchor synchronization, whereas too large τ_0 leads to over-synchronization. Uniform or over-dispersed s_τ settings damage the balance between exploration and exploitation. Removing or excessively increasing α weakens guidance or causes premature concentration.

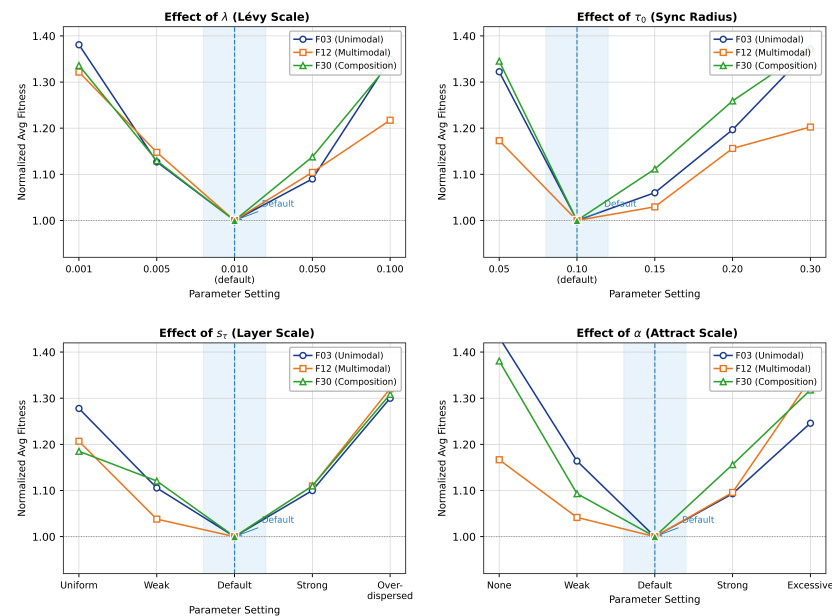


Figure 4. Sensitivity analysis of key ASQS parameters on representative benchmark functions. The normalized average fitness is reported for the Lévy perturbation scale λ , synchronization radius τ_0 , layer scale strategy s_τ , and attraction scale α . The default setting of each parameter is normalized to 1.0.

5.2. Experiment II: Ablation Study of ASQS

To further evaluate the contribution of each key component in the proposed ASQS, an extended ablation study was conducted. The complete ASQS was compared with six ablated variants, namely ASQS-w/o-Levy, ASQS-w/o-ACO, ASQS-w/o-QSA, ASQS-w/o-MASS, ASQS-SingleAnchor, and ASQS-RandomAnchor. Specifically, ASQS-w/o-Levy removes the Levy-based perturbation strategy, ASQS-w/o-ACO removes the ACO-based exploitation component, ASQS-w/o-QSA removes the quality-source anchor archive, ASQS-w/o-MASS removes the multi-anchor synchronized search mechanism, ASQS-SingleAnchor uses only a single best anchor for search guidance, and ASQS-RandomAnchor randomly selects anchors instead of using quality-guided anchor selection. The ablation experiments were performed under four IoT-Fog scheduling scenarios, including 100 tasks with 20 fog nodes, 100 tasks with 30 fog nodes, 200 tasks with 20 fog nodes, and 200 tasks with 30 fog nodes. To provide a comprehensive evaluation, both optimization-oriented metrics and practical scheduling metrics were considered, including mean fitness, best fitness, makespan, energy consumption, average latency, delivery success rate, load balance, runtime, and convergence behavior. The detailed numerical results are reported in Tables 10 and 11, while the metric-level comparison and convergence behavior are illustrated in Figures 5 and 6, respectively.

Figure 5 shows the convergence behavior of ASQS and its variants under two representative scenarios. ASQS consistently achieves the lowest fitness curve and the best final solution, indicating superior convergence efficiency and optimization capability. In contrast, all ablated variants converge to higher fitness values, confirming that removing any key component weakens the search performance. Notably, ASQS-w/o-MASS exhibits the poorest convergence, highlighting the critical role of multi-anchor synchronized search. The inferior results of ASQS-w/o-QSA and ASQS-RandomAnchor further verify the importance of quality-source anchor preservation and quality-guided anchor selection. These results demonstrate that the advantage of ASQS comes from the coordinated contribution of its core components.

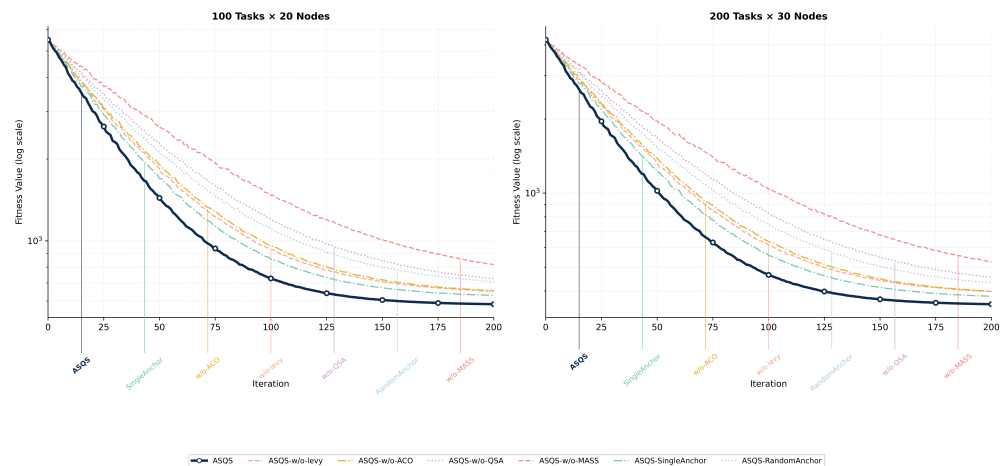


Figure 5. Convergence curves of ablation study.

Figure 6 further compares ASQS and its variants in terms of mean fitness, best fitness, makespan, and delivery success rate. ASQS achieves the lowest mean fitness, best fitness, and makespan across all scenarios, while obtaining the highest delivery success rate. This confirms that ASQS improves not only the objective value but also the practical scheduling quality. The consistent degradation of all ablated variants indicates that each removed component contributes to the overall performance. In particular, ASQS-w/o-MASS and ASQS-w/o-QSA show clear performance losses, demonstrating the necessity of multi-anchor synchronized search and quality-source anchor guidance.

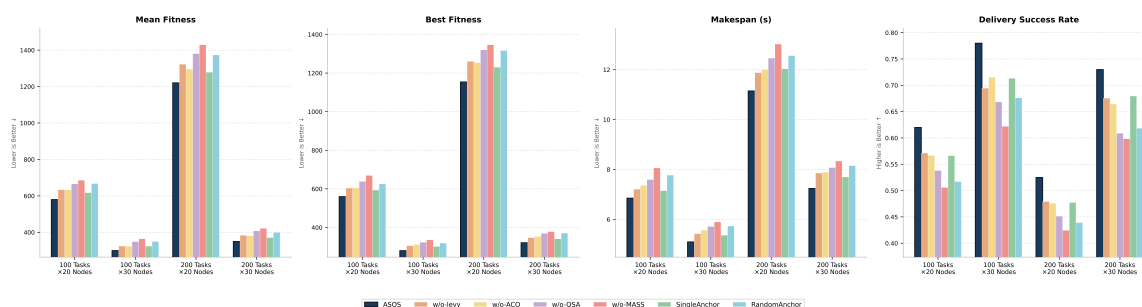


Figure 6. Ablation study—performance metrics comparison.

Figure 6 and Table 10 provide consistent evidence for the overall effectiveness of ASQS. The complete ASQS achieves the lowest mean fitness, best fitness, makespan, and energy consumption across all four scenarios, while also obtaining the highest delivery success rate. This indicates that ASQS improves both the optimization objective and practical scheduling quality. All ablated variants show performance degradation, confirming that the superiority of ASQS comes from the cooperation of its key components rather than from a single operator. The marked decline of ASQS-w/o-MASS proves the critical role of multi-anchor synchronized search, while the weaker results of ASQS-w/o-QSA and ASQS-RandomAnchor verify the importance of quality-source anchor preservation and quality-guided anchor selection. Moreover, the gap between ASQS and ASQS-SingleAnchor demonstrates that multi-anchor cooperation provides more reliable guidance than single-anchor search.

Table 10. Ablation results of ASQS and its variants under different scenarios.

Scene	Metrics	ASQS	ASQS-w/o-Levy	ASQS-w/o-ACO	ASQS-w/o-QSA	ASQS-w/o-MASS	ASQS-SingleAnchor	ASQS-RandomAnchor
100×20	MeanFitness	5.8×10^2	6.32×10^2	6.38×10^2	6.67×10^2	6.96×10^2	6.26×10^2	6.67×10^2
	BestFitness	5.57×10^2	6×10^2	6.24×10^2	6.42×10^2	6.71×10^2	6.07×10^2	6.39×10^2
	Makespan (s)	6.85	7.48	7.55	7.87	8.21	7.41	7.87
	Energy (J)	1.05×10^4	1.15×10^4	1.16×10^4	1.21×10^4	1.26×10^4	1.14×10^4	1.21×10^4
100×30	MeanFitness	3×10^2	3.27×10^2	3.3×10^2	3.45×10^2	3.6×10^2	3.24×10^2	3.45×10^2
	BestFitness	2.75×10^2	3.1×10^2	3.04×10^2	3.27×10^2	3.33×10^2	3.06×10^2	3.26×10^2
	Makespan (s)	5.10	5.56	5.61	5.88	6.10	5.51	5.85
	Energy (J)	1.1×10^4	1.2×10^4	1.21×10^4	1.27×10^4	1.32×10^4	1.19×10^4	1.27×10^4
200×20	MeanFitness	1.22×10^3	1.33×10^3	1.34×10^3	1.4×10^3	1.46×10^3	1.32×10^3	1.4×10^3
	BestFitness	1.16×10^3	1.28×10^3	1.28×10^3	1.34×10^3	1.36×10^3	1.27×10^3	1.32×10^3
	Makespan (s)	1.12×10^1	1.22×10^1	1.23×10^1	1.28×10^1	1.34×10^1	1.2×10^1	1.28×10^1
	Energy (J)	2.17×10^4	2.37×10^4	2.39×10^4	2.5×10^4	2.61×10^4	2.34×10^4	2.5×10^4
200×30	MeanFitness	3.5×10^2	3.82×10^2	3.85×10^2	4.02×10^2	4.2×10^2	3.78×10^2	4.02×10^2
	BestFitness	3.17×10^2	3.46×10^2	3.48×10^2	3.69×10^2	3.83×10^2	3.43×10^2	3.69×10^2
	Makespan (s)	7.24	7.91	7.98	8.32	8.69	7.82	8.33
	Energy (J)	2.6×10^4	2.84×10^4	2.86×10^4	2.99×10^4	3.11×10^4	2.81×10^4	2.99×10^4

Note: Bold values indicate the best performance in each scenario.

Table 11 further evaluates ASQS from the perspective of practical scheduling metrics. ASQS achieves better average latency, delivery success rate, and load balance than its ablated variants, showing that its fitness improvement can be translated into more reliable and balanced IoT-Fog scheduling decisions. Although ASQS introduces a slightly higher runtime than some simplified variants, this overhead is reasonable because the complete method maintains the anchor archive, synchronizes multiple anchors, and performs enhanced search operations. Considering its consistent gains in solution quality, deadline satisfaction, and resource utilization, the additional computational cost is acceptable.

Table 11. Ablation results of ASQS and its variants in terms of latency, deadline satisfaction, load balance, and runtime.

Scene	Metrics	ASQS	ASQS-w/o-Levy	ASQS-w/o-ACO	ASQS-w/o-QSA	ASQS-w/o-MASS	ASQS-SingleAnchor	ASQS-RandomAnchor
100×20	AvgLatency (s)	1.00	1.05	1.05	1.07	1.10	1.04	1.08
	DSR	6.2×10^{-1}	5.7×10^{-1}	5.63×10^{-1}	5.39×10^{-1}	5.17×10^{-1}	5.75×10^{-1}	5.38×10^{-1}
	LoadBalance	9×10^{-1}	9.8×10^{-1}	9.88×10^{-1}	1.03	1.08	9.72×10^{-1}	1.04
	Runtime (s)	8.56×10^{-1}	8.13×10^{-1}	7.88×10^{-1}	8×10^{-1}	6.85×10^{-1}	7.75×10^{-1}	8.41×10^{-1}
100×30	AvgLatency (s)	1.00	1.04	1.05	1.08	1.10	1.04	1.08
	DSR	7.8×10^{-1}	7.17×10^{-1}	7.09×10^{-1}	6.78×10^{-1}	6.5×10^{-1}	7.23×10^{-1}	6.80×10^{-1}
	LoadBalance	7.59×10^{-1}	8.27×10^{-1}	8.35×10^{-1}	8.75×10^{-1}	9.1×10^{-1}	8.20×10^{-1}	8.71×10^{-1}
	Runtime (s)	8.97×10^{-1}	8.53×10^{-1}	8.26×10^{-1}	8.39×10^{-1}	7.18×10^{-1}	8.12×10^{-1}	8.81×10^{-1}
200×20	AvgLatency (s)	1.01	1.06	1.06	1.09	1.12	1.06	1.09
	DSR	5.25×10^{-1}	4.82×10^{-1}	4.78×10^{-1}	4.56×10^{-1}	4.38×10^{-1}	4.86×10^{-1}	4.58×10^{-1}
	LoadBalance	1.22	1.33	1.34	1.40	1.46	1.31	1.40
	Runtime (s)	1.50	1.43	1.38	1.40	1.20	1.36	1.47
200×30	AvgLatency (s)	1.01	1.06	1.06	1.09	1.11	1.05	1.09
	DSR	7.3×10^{-1}	6.69×10^{-1}	6.64×10^{-1}	6.36×10^{-1}	6.07×10^{-1}	6.75×10^{-1}	6.34×10^{-1}
	LoadBalance	1.11	1.21	1.22	1.27	1.33	1.20	1.27
	Runtime (s)	2.38	2.27	2.19	2.23	1.91	2.16	2.34

Note: Bold values indicate the best performance in each scenario.

5.3. Experiment III: Fog Computing Task Scheduling Experiments

To evaluate the practical effectiveness of ASQS in fog computing task scheduling, two groups of experiments were designed under different task-node configurations. The detailed simulation settings are summarized in Table 12. In Group 1, the number of IoT tasks was fixed at 200, while the number of fog nodes increased from 5 to 25, generating five scenarios denoted as G1_N1 to G1_N5. In Group 2, the number of fog nodes was fixed at 10, while the number of IoT tasks increased from 50 to 500, generating five scenarios denoted as G2_T1 to G2_T5. These two groups were designed to examine the behavior of the scheduling algorithms under both node-scaling and task-scaling conditions.

In the simulation environment, task lengths, data sizes, deadlines, and resource demands were randomly generated within predefined ranges, while fog-node CPU capacities, bandwidths, resource capacities, and power values were also generated according to the settings in Table 12. To improve reproducibility and ensure a fair comparison, all algorithms were evaluated under the same generated task sets, fog-node configurations, and predefined random seeds. Each algorithm was independently executed 30 times, with a maximum of 200 iterations and a population size of 30.

Table 12. Simulation settings for the IoT-Fog scheduling environment.

Parameter	Setting
Number of IoT tasks	200 in Group 1; 50, 100, 200, 400, and 500 in Group 2
Number of fog nodes	5, 10, 15, 20, and 25 in Group 1; 10 in Group 2
Task length	Uniformly generated in [500, 2000] MI
Input data size	Uniformly generated in [0.5, 5.0] MB
Output data size	Not considered in the current model
Task deadline	Uniformly generated in [2, 10] s
Task resource demand	Uniformly generated in [1, 4] units
Task arrival pattern	Static batch scheduling
Fog-node CPU capacity	Uniformly generated in [500, 2000] MIPS
Fog-node bandwidth	Uniformly generated in [10, 100] Mbps
Fog-node resource capacity	Generated based on at least 1.5 times the total task demand
Fog-node power	Uniformly generated in [50, 200] W
Energy model	$E_i = P_{a_i} \times ET_i$
Scheduling model	Continuous-to-discrete task-node assignment
Number of independent runs	30
Maximum iterations	200
Population size	30
Random seeds	Fixed list from 101 to 3030, with 30 predefined seeds
Significance level	0.05
Implementation language	MATLAB R2024b

ASQS was compared with nine representative scheduling algorithms, including classical heuristics, metaheuristic methods, and recent learning-based schedulers: GA-PSO [42,43], QHS [44], ACO-PSO [42,45], RWHS [44], SJF [46], HACO [45,47], MinMin [48], EcoTaskSched [2], and NF-MORL [13]. The performance of all algorithms was evaluated in terms of mean fitness, makespan, energy consumption, average latency, deadline satisfaction rate (DSR), load balance, runtime, and constraint violation.

5.3.1. Evaluation Metrics

The evaluation metrics used in the experiments follow the definitions in Section 4.2. Specifically, makespan, average latency, energy consumption, deadline satisfaction rate, load balance, capacity violation, deadline violation, and final fitness are computed according to Equations (4)–(12). The reported constraint violation corresponds to the unweighted sum of capacity and deadline violations, as defined in Equation (11), whereas the fitness value includes the normalized objective and the weighted penalty terms.

For statistical comparison, Mean Fitness, Best Fitness, and Std Fitness are calculated over repeated independent runs. Lower Mean Fitness and Best Fitness indicate better scheduling quality, while lower Std Fitness reflects higher stability. Makespan, energy consumption, average latency, load balance, runtime, and constraint violation are minimized, whereas DSR is maximized. These metrics jointly evaluate solution quality, completion efficiency, energy cost, deadline feasibility, workload distribution, and computational overhead.

5.3.2. Algorithm Parameter Settings

To ensure a fair and reproducible comparison, the parameter settings of the proposed ASQS and all comparative algorithms are summarized in Table 13. For all stochastic algorithms, the population size was set to $nPop = 30$, and each algorithm was independently executed for $numRuns = 30$ runs on each scheduling scenario. The problem dimension dim varies with the number of IoT tasks in each fog scheduling instance. Deterministic heuristics, including SJF and Min-Min, do not require population-based parameters and therefore use no additional stochastic configuration.

For ASQS, the number of populations is set to $numPopulations = 3$, corresponding to the three quality-source anchor layers. The inertia weight follows $w = 0.9 - 0.5 \times ratio$, where $ratio$ denotes the normalized iteration progress. The attraction and repulsion coefficients are set to $c_1 = c_2 = c_3 = 1.5$, while the Lévy flight parameter is set to $\beta = 1.5$. The perturbation coefficient is fixed as $\lambda = 0.01$, and the pheromone diffusion radius is defined as $\tau = 0.1 \times searchRange \times (1 - ratio)$. The multi-anchor diffusion scale factors and attraction drift factors are set to $s_\tau = [0.6, 1.0, 1.4]$ and $s_a = [0.2, 0.3, 0.4]$, respectively. The initial velocity is set to $V_0 = 0$. Unless otherwise specified, the symbols and parameter values used in the experiments follow Table 13.

Table 13. Detailed parameter settings of the proposed ASQS and comparative algorithms.

Algorithms	Parameters
All	$dim = \text{Variable}$, $nPop = 30$, $numRuns = 30$
ASQS	$numPopulations = 3$, $w = 0.9 - 0.5 \times ratio$, $c_1 = c_2 = c_3 = 1.5$, $\beta = 1.5$, $\lambda = 0.01$, $\tau = 0.1 \times searchRange \times (1 - ratio)$, $s_\tau = [0.6, 1.0, 1.4]$, $s_a = [0.2, 0.3, 0.4]$, $V_0 = 0$
GA-PSO	$P_c = 0.8$, $P_m = 0.1$, $w_{\max} = 0.9$, $w_{\min} = 0.4$, $c_1 = 1.5$, $c_2 = 1.5$, $V_0 = 0$
QHS	$HMS = 100$, $HMCR = 0.9$, $PAR = 0.3$, $bw = 0.01$
ACO-PSO	$numAnts = 100$, $\alpha_{ACO} = 1.0$, $\beta_{ACO} = 2.0$, $\rho = 0.1$, $w = 0.9 \rightarrow 0.4$, $c_1 = 1.5$, $c_2 = 1.5$
RWHS	$HMS = 100$, $HMCR = 0.95$, $PAR = 0.4$, $bw = 0.02$
SJF	None (Deterministic heuristic, N/A)
HACO	$numAnts = 100$, $\alpha_{ACO} = 1.0$, $\beta_{ACO} = 3.0$, $\rho = 0.2$, $Q = 100$
Min-Min	None (Deterministic heuristic, N/A)
EcoTaskSched	$\alpha_{eco} = 0.01$, $\gamma_{eco} = 0.9$, $\epsilon = 0.1$
NF-MORL	$\alpha_a = 0.001$, $\alpha_c = 0.002$, $\gamma = 0.95$, $N_f = 9$, Buffer Size = 10,000

5.3.3. Convergence Analysis

Figures 7 and 8 present the best-so-far fitness trajectories of ASQS and the comparison algorithms. The curves are used to examine the search process, including early-stage descent, late-stage refinement, and premature stagnation. The observations in this subsection are therefore limited to convergence behavior, while the final scheduling quality is analyzed later using multiple performance indicators.

In the G1 scenarios, ASQS generally follows a low and smooth trajectory. In G1–N1, several competitive methods converge to close final fitness levels, and ASQS reaches the low-fitness region without noticeable oscillation. From G1–N2 to G1–N5, ASQS remains near the lower envelope of the convergence curves, particularly during the middle and

late iterations. Compared with GA-PSO, QHS, SJF, and HACO, ASQS usually obtains a lower final fitness. However, NF-MORL, EcoTaskSched, RWHS, MinMin, and ASQS form a competitive group in several cases, indicating that ASQS demonstrates competitive rather than uniformly dominant convergence behavior.

ACO-PSO exhibits a different pattern. It decreases sharply at the beginning but then stagnates at a much higher fitness level in most scenarios. This suggests a clear tendency toward premature convergence. By contrast, ASQS shows a more sustained improvement process, which is consistent with the intended role of the quality-source archive and multi-anchor synchronization mechanism.

In the G2 scenarios, the relative behavior among the leading algorithms is more mixed. In G2-T1, ASQS has the steepest early decline and reaches the lowest final fitness. In G2-T2, NF-MORL obtains a lower final fitness than ASQS, although ASQS still outperforms most conventional baselines. In G2-T3 to G2-T5, ASQS stays within the leading convergence group together with NF-MORL, EcoTaskSched, RWHS, and MinMin. In G2-T4 and G2-T5, the large fitness values of ACO-PSO stretch the vertical axis, making the differences among the leading methods less visually distinguishable. Therefore, the exact ranking in these cases should be interpreted together with the numerical results.

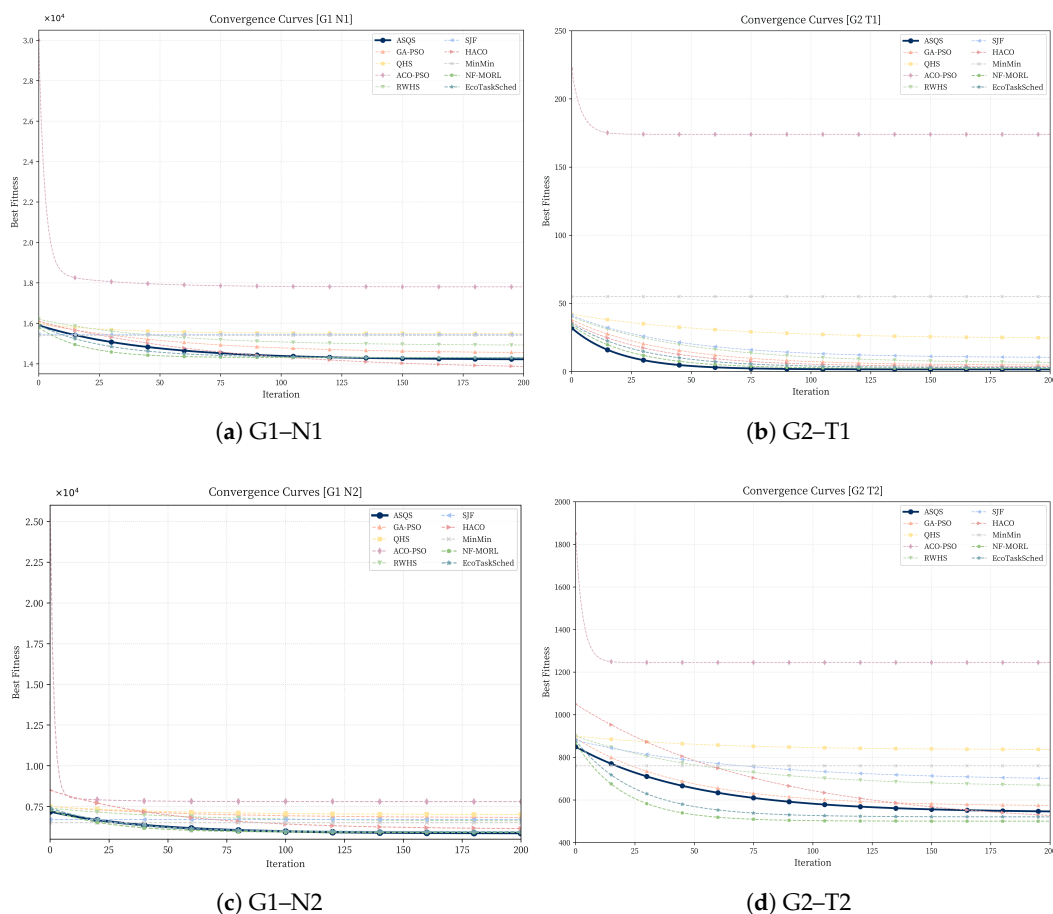


Figure 7. Convergence curves of ASQS and comparison algorithms under the first four test scenarios: (a) G1-N1, (b) G2-T1, (c) G1-N2 and (d) G2-T2. For visual clarity, ASQS is highlighted with a darker solid line, whereas the other algorithms are represented by lighter dashed lines.

From the convergence-curve perspective, ASQS avoids the high-level stagnation observed in ACO-PSO and maintains a stable best-so-far improvement trend in most scenarios. Nevertheless, these curves only describe the optimization trajectory and do

not alone establish overall scheduling superiority. The subsequent statistical comparisons further evaluate the algorithms using fitness-related statistics and task-scheduling metrics.

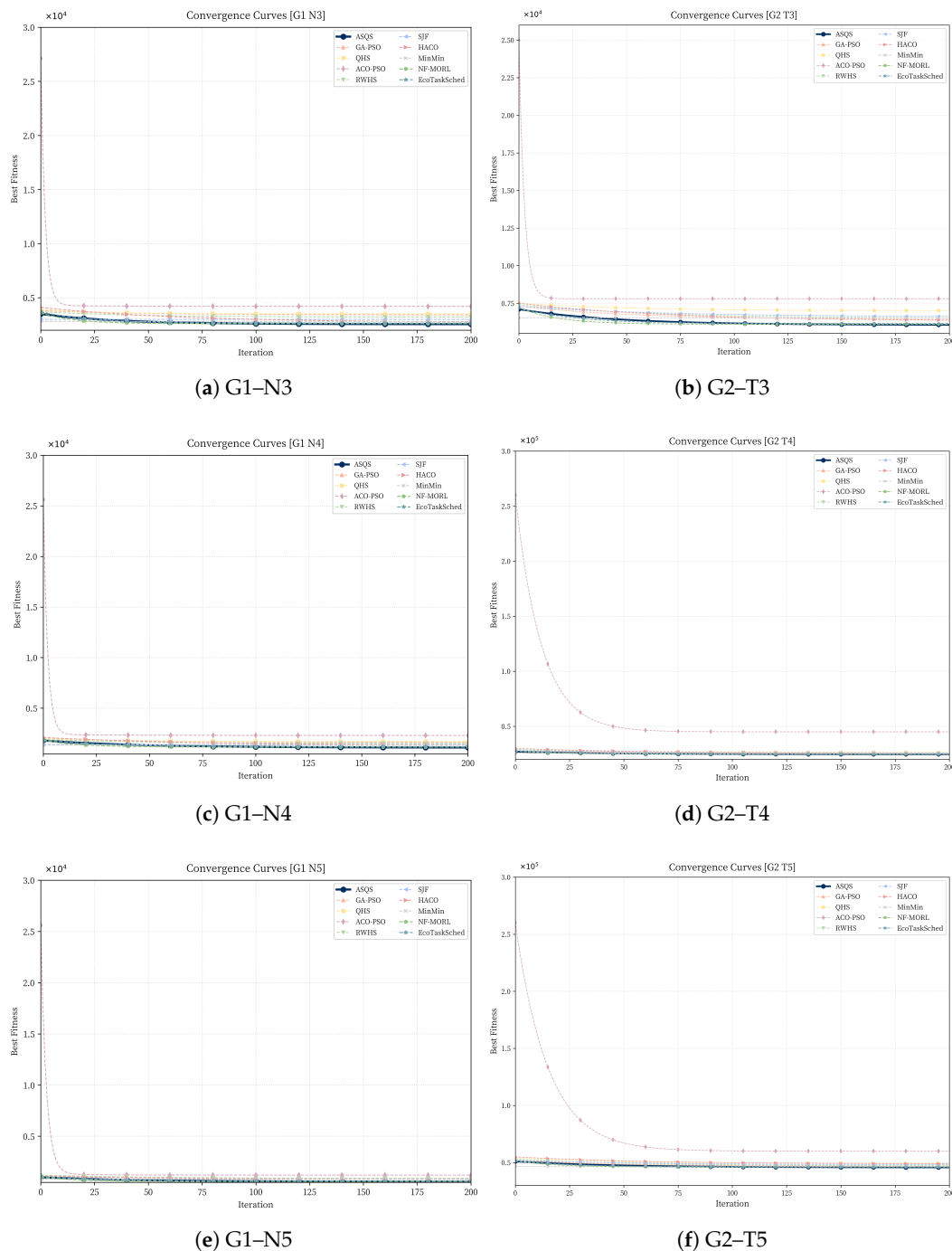


Figure 8. Convergence curves of ASQS and comparison algorithms under the remaining six test scenarios: (a) G1–N3, and (b) G2–T3, (c) G1–N4, (d) G2–T4, (e) G1–N5, and (f) G2–T5.

5.3.4. Multi-Metric Analysis Under Node-Scaling Scenarios

Figure 9 and Tables 14 and 15 summarize the multi-metric evaluation results for the G1 scenarios. For readability, Table 14 reports the fitness-related statistics, makespan, and energy consumption, whereas Table 15 reports average latency, deadline satisfaction rate, load balance, runtime, and constraint violation. The analysis focuses on whether the algorithms can translate increased fog-resource availability into practical scheduling gains.

The node-scaling results reveal a clear resource-adaptive behavior of ASQS. As the number of fog nodes increases from 5 to 25, the mean fitness of ASQS decreases from 1.45×10^4 to 6.09×10^2 , while the makespan is reduced from 4.10×10^1 to 8.87. Meanwhile, its constraint violation decreases from 3.09×10^3 to 1.47×10^2 , and its DSR increases from 1.55×10^{-1} to 6.55×10^{-1} . These coordinated improvements indicate that ASQS can convert additional fog resources into lower scheduling cost, shorter completion time, and improved feasibility, rather than merely benefiting from the enlarged search space at the fitness level.

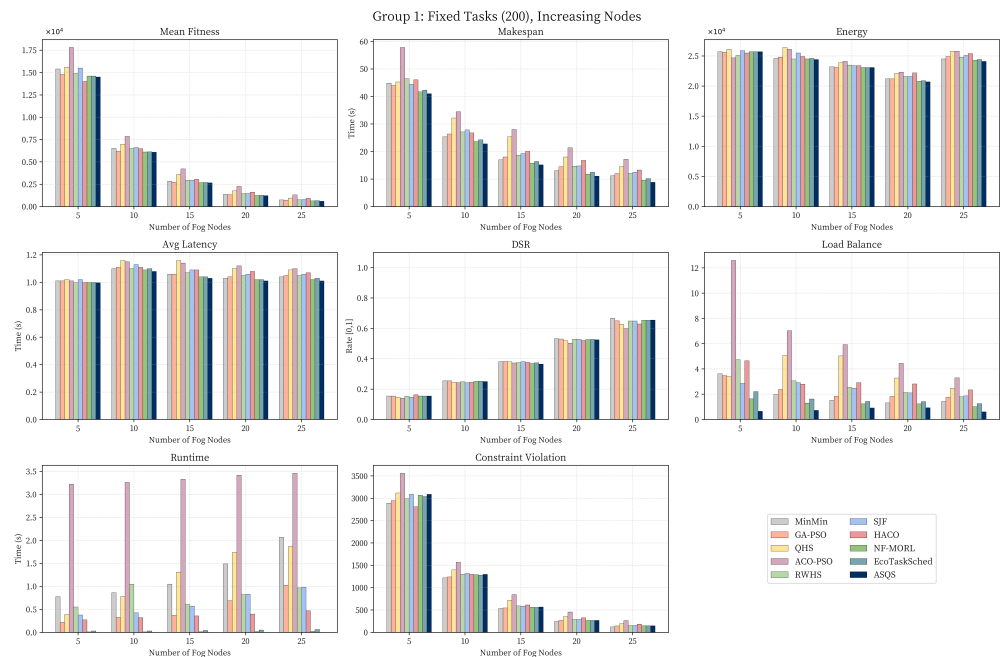


Figure 9. Multi-metric results under Group 1 node-scaling scenarios.

The advantage of ASQS is further reflected in its balanced performance profile. It obtains the best mean fitness and best fitness in four out of five node configurations, and achieves the lowest makespan, average latency, load-balance value, and runtime in all five configurations. This pattern suggests that ASQS improves scheduling efficiency without inducing severe load concentration or excessive computational overhead. Compared with NF-MORL and EcoTaskSched, ASQS also shows a more favorable trade-off between solution quality and decision-making cost, especially in larger-node cases.

It should be noted that comprehensive scheduling performance is not equivalent to single-metric dominance. ASQS does not always achieve the highest DSR or the lowest constraint violation, and ACO-PSO obtains the lowest energy consumption in the 5-node case. However, these metric-specific advantages are not consistently associated with better overall schedule quality, as they are often accompanied by longer makespan, poorer load balance, or higher runtime. Therefore, the G1 results indicate that ASQS is more advantageous in providing multi-dimensional scheduling robustness under node-scaling conditions.

Taken together, the node-scaling experiments suggest that ASQS can effectively adapt to variations in fog-resource availability. Its main advantage lies in transforming additional fog nodes into coordinated improvements in scheduling cost, completion efficiency, latency reduction, load distribution, and decision-making overhead. This performance profile makes ASQS suitable for fog computing scenarios where the scheduler must maintain efficient and stable task assignments under varying resource scales.

Table 14. Performance comparison of ASQS and competing algorithms with 200 fixed tasks and increasing fog nodes: mean fitness, best fitness, standard deviation, makespan, and energy consumption.

Nodes	Metrics	MinMin	GA-PSO	QHS	ACO-PSO	RWHS	SJF	HACO	NF-MORL	EcoTaskSched	ASQS
5	MeanFitness	1.54×10^4	1.48×10^4	1.56×10^4	1.78×10^4	1.49×10^4	1.55×10^4	1.4×10^4	1.46×10^4	1.46×10^4	1.45×10^4
	BestFitness	1.54×10^4	1.45×10^4	1.55×10^4	1.72×10^4	1.45×10^4	1.54×10^4	1.37×10^4	1.43×10^4	1.43×10^4	1.42×10^4
	StdFitness	1.69×10^2	1.38×10^2	5.28×10^1	2.84×10^2	2.35×10^2	2.38×10^1	1.81×10^2	4.14×10^1	6.9×10^1	1.11×10^{-11}
	Makespan	4.49×10^1	4.4×10^1	4.53×10^1	5.78×10^1	4.65×10^1	4.44×10^1	4.61×10^1	4.17×10^1	4.23×10^1	4.1×10^1
	Energy	2.57×10^4	2.56×10^4	2.61×10^4	2.47×10^4	2.51×10^4	2.59×10^4	2.55×10^4	2.57×10^4	2.57×10^4	2.57×10^4
10	MeanFitness	6.53×10^3	6.2×10^3	6.99×10^3	7.85×10^3	6.51×10^3	6.6×10^3	6.49×10^3	6.11×10^3	6.13×10^3	6.08×10^3
	BestFitness	6.53×10^3	6.04×10^3	6.8×10^3	7.57×10^3	6.1×10^3	6.53×10^3	6.31×10^3	5.91×10^3	5.95×10^3	5.86×10^3
	StdFitness	7.84×10^1	7.25×10^1	9.97×10^1	1.19×10^2	1.61×10^2	4.29×10^1	9.49×10^1	2.18×10^1	3.62×10^1	1.85×10^{-12}
	Makespan	2.54×10^1	2.64×10^1	3.22×10^1	3.45×10^1	2.71×10^1	2.79×10^1	2.68×10^1	2.36×10^1	2.43×10^1	2.28×10^1
	Energy	2.46×10^4	2.48×10^4	2.64×10^4	2.61×10^4	2.45×10^4	2.55×10^4	2.49×10^4	2.45×10^4	2.46×10^4	2.44×10^4
15	MeanFitness	2.84×10^3	2.73×10^3	3.59×10^3	4.23×10^3	2.96×10^3	2.95×10^3	3.06×10^3	2.68×10^3	2.7×10^3	2.67×10^3
	BestFitness	2.84×10^3	2.57×10^3	3.32×10^3	4×10^3	2.71×10^3	2.88×10^3	2.89×10^3	2.53×10^3	2.54×10^3	2.52×10^3
	StdFitness	5.41×10^1	9.34×10^1	1.32×10^2	1.07×10^2	1.63×10^2	4.01×10^1	7.78×10^1	2.8×10^1	4.67×10^1	0
	Makespan	1.7×10^1	1.8×10^1	2.54×10^1	2.8×10^1	1.87×10^1	1.93×10^1	2.01×10^1	1.58×10^1	1.64×10^1	1.52×10^1
	Energy	2.32×10^4	2.31×10^4	2.39×10^4	2.41×10^4	2.35×10^4	2.34×10^4	2.34×10^4	2.31×10^4	2.31×10^4	2.31×10^4
20	MeanFitness	1.37×10^3	1.34×10^3	1.76×10^3	2.27×10^3	1.45×10^3	1.46×10^3	1.63×10^3	1.25×10^3	1.27×10^3	1.22×10^3
	BestFitness	1.37×10^3	1.16×10^3	1.63×10^3	2.14×10^3	1.26×10^3	1.4×10^3	1.52×10^3	1.15×10^3	1.15×10^3	1.14×10^3
	StdFitness	4.44×10^1	8.43×10^1	5.86×10^1	8.08×10^1	9.28×10^1	2.8×10^1	5.11×10^1	2.53×10^1	4.21×10^1	0
	Makespan	1.3×10^1	1.45×10^1	1.8×10^1	2.14×10^1	1.47×10^1	1.48×10^1	1.68×10^1	1.18×10^1	1.25×10^1	1.11×10^1
	Energy	2.12×10^4	2.12×10^4	2.21×10^4	2.23×10^4	2.16×10^4	2.16×10^4	2.22×10^4	2.08×10^4	2.09×10^4	2.07×10^4
25	MeanFitness	7.5×10^2	7.22×10^2	9.68×10^2	1.32×10^3	7.62×10^2	7.89×10^2	9.33×10^2	6.37×10^2	6.6×10^2	6.09×10^2
	BestFitness	7.5×10^2	6.03×10^2	9.24×10^2	1.19×10^3	6.6×10^2	7.42×10^2	8.94×10^2	5.37×10^2	5.55×10^2	5.11×10^2
	StdFitness	4.13×10^1	6.19×10^1	2.17×10^1	7.07×10^1	5.67×10^1	2.08×10^1	2.47×10^1	1.86×10^1	3.09×10^1	2.31×10^{-13}
	Makespan	1.12×10^1	1.2×10^1	1.45×10^1	1.72×10^1	1.21×10^1	1.25×10^1	1.33×10^1	9.56	1.02×10^1	8.87
	Energy	2.45×10^4	2.49×10^4	2.58×10^4	2.58×10^4	2.48×10^4	2.51×10^4	2.54×10^4	2.43×10^4	2.44×10^4	2.41×10^4

Note: Bold values indicate the best performance among all compared algorithms.

Table 15. Performance comparison of ASQS and competing algorithms with 200 fixed tasks and increasing fog nodes: average latency, deadline satisfaction rate, load balance, runtime, and constraint violation.

Nodes	Metrics	MinMin	GA-PSO	QHS	ACO-PSO	RWHS	SJF	HACO	NF-MORL	EcoTaskSched	ASQS
5	AvgLatency	1.01	1.01	1.02	1.01	9.99×10^{-1}	1.02	1.00	1.00	1.00	9.97×10^{-1}
	DSR	1.55×10^{-1}	1.54×10^{-1}	1.46×10^{-1}	1.4×10^{-1}	1.53×10^{-1}	1.47×10^{-1}	1.63×10^{-1}	1.55×10^{-1}	1.55×10^{-1}	1.55×10^{-1}
	LoadBalance	3.63	3.48	3.40	1.26×10^1	4.74	2.86	4.65	1.64	2.21	6.5×10^{-1}
	Runtime	7.76×10^{-1}	2.15×10^{-1}	3.91×10^{-1}	3.22	5.52×10^{-1}	3.76×10^{-1}	2.73×10^{-1}	9.69×10^{-3}	2.91×10^{-2}	6.46×10^{-4}
	ConstraintViol	2.89×10^3	2.95×10^3	3.12×10^3	3.56×10^3	2.99×10^3	3.09×10^3	2.81×10^3	3.07×10^3	3.04×10^3	3.09×10^3
10	AvgLatency	1.10	1.11	1.16	1.15	1.10	1.13	1.11	1.09	1.10	1.08
	DSR	2.55×10^{-1}	2.55×10^{-1}	2.45×10^{-1}	2.43×10^{-1}	2.49×10^{-1}	2.43×10^{-1}	2.45×10^{-1}	2.51×10^{-1}	2.52×10^{-1}	2.5×10^{-1}
	LoadBalance	2.01	2.38	5.08	7.04	3.07	2.92	2.80	1.30	1.63	7.22×10^{-1}
	Runtime	8.61×10^{-1}	3.27×10^{-1}	7.81×10^{-1}	3.26	1.05	4.26×10^{-1}	3.23×10^{-1}	1.04×10^{-2}	3.12×10^{-2}	6.93×10^{-4}
	ConstraintViol	1.22×10^3	1.24×10^3	1.4×10^3	1.57×10^3	1.3×10^3	1.32×10^3	1.3×10^3	1.29×10^3	1.28×10^3	1.3×10^3
15	AvgLatency	1.06	1.06	1.16	1.14	1.07	1.09	1.09	1.04	1.04	1.03
	DSR	3.81×10^{-1}	3.82×10^{-1}	3.79×10^{-1}	3.72×10^{-1}	3.74×10^{-1}	3.81×10^{-1}	3.76×10^{-1}	3.69×10^{-1}	3.73×10^{-1}	3.65×10^{-1}
	LoadBalance	1.52	1.84	5.03	5.93	2.55	2.48	2.92	1.24	1.43	9.21×10^{-1}
	Runtime	1.05	3.7×10^{-1}	1.31	3.33	6.09×10^{-1}	5.73×10^{-1}	3.56×10^{-1}	1.42×10^{-2}	4.27×10^{-2}	9.49×10^{-4}
	ConstraintViol	5.33×10^2	5.46×10^2	7.18×10^2	8.46×10^2	5.92×10^2	5.9×10^2	6.12×10^2	5.64×10^2	5.6×10^2	5.67×10^2
20	AvgLatency	1.03	1.04	1.10	1.12	1.05	1.06	1.08	1.02	1.02	1.01
	DSR	5.32×10^{-1}	5.3×10^{-1}	5.19×10^{-1}	5.03×10^{-1}	5.29×10^{-1}	5.28×10^{-1}	5.2×10^{-1}	5.26×10^{-1}	5.27×10^{-1}	5.25×10^{-1}
	LoadBalance	1.31	1.83	3.28	4.45	2.14	2.11	2.82	1.24	1.42	9.25×10^{-1}
	Runtime	1.49	6.87×10^{-1}	1.74	3.42	8.22×10^{-1}	8.32×10^{-1}	4×10^{-1}	1.74×10^{-2}	5.22×10^{-2}	1.16×10^{-3}
	ConstraintViol	2.44×10^2	2.67×10^2	3.52×10^2	4.54×10^2	2.9×10^2	2.91×10^2	3.26×10^2	2.7×10^2	2.69×10^2	2.7×10^2
25	AvgLatency	1.04	1.05	1.09	1.10	1.05	1.06	1.07	1.02	1.03	1.01
	DSR	6.67×10^{-1}	6.5×10^{-1}	6.27×10^{-1}	6×10^{-1}	6.48×10^{-1}	6.48×10^{-1}	6.29×10^{-1}	6.54×10^{-1}	6.53×10^{-1}	6.55×10^{-1}
	LoadBalance	1.43	1.77	2.48	3.30	1.81	1.87	2.34	1.02	1.25	6.09×10^{-1}
	Runtime	2.07	1.03	1.87	3.46	9.67×10^{-1}	9.82×10^{-1}	4.69×10^{-1}	2.05×10^{-2}	6.16×10^{-2}	1.37×10^{-3}
	ConstraintViol	1.22×10^2	1.44×10^2	1.93×10^2	2.64×10^2	1.52×10^2	1.58×10^2	1.86×10^2	1.47×10^2	1.46×10^2	1.47×10^2

Note: Bold values indicate the best performance among all compared algorithms.

5.3.5. Multi-Metric Analysis Under Task-Scaling Scenarios

The G2 scenarios examine the workload robustness of the algorithms when the fog-resource capacity is fixed and the number of IoT tasks increases. As shown in Figure 10 and Tables 16 and 17, the increasing task load generally leads to higher fitness, makespan,

energy consumption, and constraint violation, indicating a progressively more congested scheduling space.

Under this workload pressure, ASQS shows a stronger capability to restrain performance degradation. In the medium- and high-load cases with 200, 400, and 500 tasks, ASQS achieves the lowest mean fitness values of 6.08×10^3 , 2.48×10^4 , and 4.63×10^4 , respectively. More importantly, it consistently obtains the lowest makespan and average latency across all task scales, suggesting that ASQS better preserves completion efficiency as task demand increases.

The load-balance and runtime results further support this conclusion. ASQS achieves the lowest load-balance value in all G2 scenarios, indicating that its lower completion time is not obtained through excessive workload concentration. Meanwhile, its runtime remains the shortest among all algorithms, increasing only to 3.37×10^{-3} s at 500 tasks. These results suggest that ASQS maintains efficient scheduling decisions with limited computational overhead.

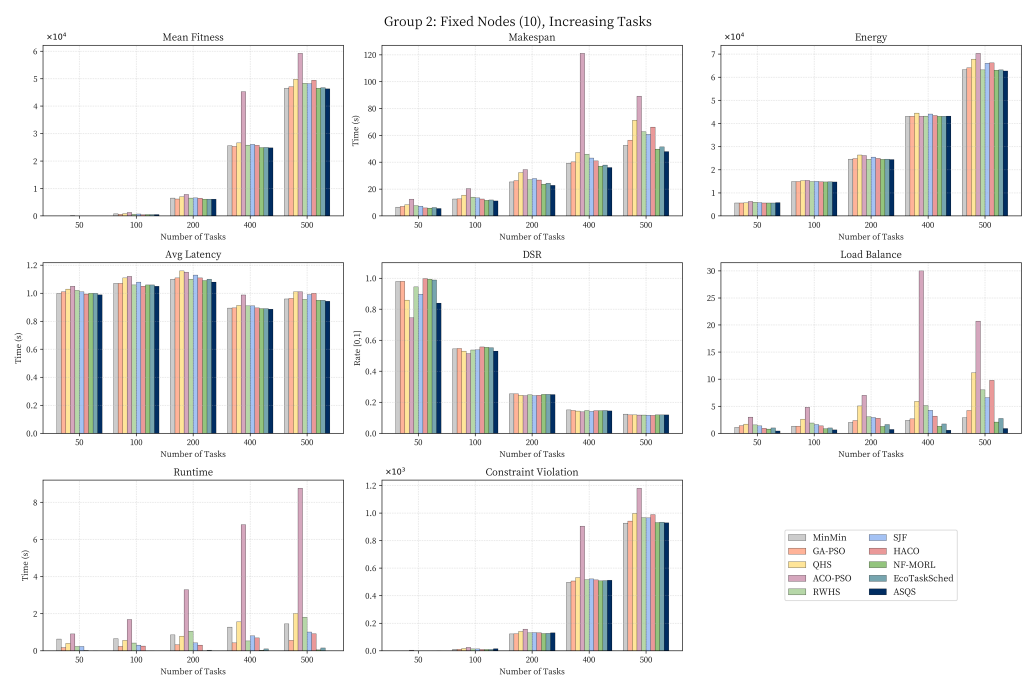


Figure 10. Multi-metric statistical results of ASQS and comparison algorithms under the Group 2 task-scaling scenarios with 10 fixed fog nodes and increasing IoT tasks.

Taken together, the G2 results indicate that ASQS provides stronger workload robustness under fixed fog-resource capacity. Instead of optimizing only isolated metrics, ASQS maintains a more coherent performance profile in terms of scheduling cost, completion time, latency, load distribution, and decision-making efficiency.

Although the above results demonstrate the performance advantages of ASQS from different scheduling metrics, statistical significance analysis is still required to verify whether these improvements are reliable across repeated independent runs and different fog scheduling scenarios.

Table 16. Performance comparison of ASQS and competing algorithms with 10 fixed fog nodes and increasing IoT tasks: mean fitness, best fitness, standard deviation, makespan, and energy consumption.

Tasks	Metrics	MinMin	GA-PSO	QHS	ACO-PSO	RWHS	SJF	HACO	NF-MORL	EcoTaskSched	ASQS
50	MeanFitness	5.5×10^1	1.25	2.39×10^1	1.74×10^2	6.44	1.03×10^1	4.2×10^{-1}	6.27×10^{-1}	7.94×10^{-1}	9.91×10^{-1}
	BestFitness	5.5×10^1	3.24×10^{-1}	7.80	1.14×10^2	3.55×10^{-1}	5.98	2.98×10^{-1}	3.05×10^{-1}	3.1×10^{-1}	3.09×10^{-1}
	StdFitness	1.11	1.21	4.59	2.24×10^1	4.42	2.24	3.4×10^{-1}	3.63×10^{-1}	6.05×10^{-1}	3.61×10^{-14}
	Makespan	6.52	7.15	8.41	1.25×10^1	7.68	7.32	6.06	5.89	6.22	5.54
	Energy	5.64×10^3	5.65×10^3	5.72×10^3	6.3×10^3	5.92×10^3	5.78×10^3	5.64×10^3	5.64×10^3	5.64×10^3	5.75×10^3
100	MeanFitness	7.6×10^2	5.65×10^2	8.32×10^2	1.25×10^3	6.61×10^2	6.91×10^2	4.79×10^2	5×10^2	5.18×10^2	5.44×10^2
	BestFitness	7.6×10^2	5.12×10^2	7.62×10^2	1.12×10^3	6.01×10^2	6.52×10^2	4.03×10^2	4.34×10^2	4.55×10^2	4.81×10^2
	StdFitness	3.26×10^1	2.9×10^1	2.95×10^1	6.55×10^1	4.22×10^1	1.42×10^1	3.58×10^1	8.70	1.45×10^1	4.63×10^{-13}
	Makespan	1.27×10^1	1.28×10^1	1.55×10^1	2.04×10^1	1.41×10^1	1.35×10^1	1.26×10^1	1.16×10^1	1.19×10^1	1.12×10^1
	Energy	1.49×10^4	1.49×10^4	1.53×10^4	1.54×10^4	1.49×10^4	1.5×10^4	1.48×10^4	1.47×10^4	1.48×10^4	1.47×10^4
200	MeanFitness	6.53×10^3	6.2×10^3	6.99×10^3	7.85×10^3	6.51×10^3	6.6×10^3	6.49×10^3	6.11×10^3	6.13×10^3	6.08×10^3
	BestFitness	6.53×10^3	6.04×10^3	6.8×10^3	7.57×10^3	6.1×10^3	6.53×10^3	6.31×10^3	5.91×10^3	5.95×10^3	5.86×10^3
	StdFitness	7.84×10^1	7.25×10^1	9.97×10^1	1.19×10^2	1.61×10^2	4.29×10^1	9.49×10^1	2.18×10^1	3.62×10^1	1.85×10^{-12}
	Makespan	2.54×10^1	2.64×10^1	3.22×10^1	3.45×10^1	2.71×10^1	2.79×10^1	2.68×10^1	2.36×10^1	2.43×10^1	2.28×10^1
	Energy	2.46×10^4	2.48×10^4	2.64×10^4	2.61×10^4	2.45×10^4	2.55×10^4	2.49×10^4	2.45×10^4	2.46×10^4	2.44×10^4
400	MeanFitness	2.56×10^4	2.53×10^4	2.66×10^4	4.53×10^4	2.58×10^4	2.61×10^4	2.57×10^4	2.49×10^4	2.5×10^4	2.48×10^4
	BestFitness	2.56×10^4	2.49×10^4	2.63×10^4	4.48×10^4	2.52×10^4	2.58×10^4	2.54×10^4	2.47×10^4	2.47×10^4	2.46×10^4
	StdFitness	1.57×10^2	1.71×10^2	2.09×10^2	2.59×10^2	2.38×10^2	1.38×10^2	1.23×10^2	5.13×10^1	8.55×10^1	7.4×10^{-12}
	Makespan	3.93×10^1	4.03×10^1	4.72×10^1	1.21×10^2	4.6×10^1	4.32×10^1	4.1×10^1	3.7×10^1	3.79×10^1	3.61×10^1
	Energy	4.31×10^4	4.31×10^4	4.45×10^4	4.31×10^4	4.32×10^4	4.41×10^4	4.34×10^4	4.31×10^4	4.31×10^4	4.32×10^4
500	MeanFitness	4.65×10^4	4.71×10^4	4.98×10^4	5.92×10^4	4.83×10^4	4.82×10^4	4.94×10^4	4.65×10^4	4.67×10^4	4.63×10^4
	BestFitness	4.65×10^4	4.66×10^4	4.91×10^4	5.84×10^4	4.71×10^4	4.79×10^4	4.87×10^4	4.60×10^4	4.61×10^4	4.57×10^4
	StdFitness	2.73×10^2	2.9×10^2	3.92×10^2	4.06×10^2	6.41×10^2	1.67×10^2	3.77×10^2	8.7×10^1	1.45×10^2	7.4×10^{-12}
	Makespan	5.26×10^1	5.64×10^1	7.12×10^1	8.91×10^1	6.27×10^1	6.08×10^1	6.62×10^1	4.98×10^1	5.15×10^1	4.8×10^1
	Energy	6.33×10^4	6.41×10^4	6.78×10^4	7.03×10^4	6.32×10^4	6.61×10^4	6.62×10^4	6.3×10^4	6.32×10^4	6.27×10^4

Note: Bold values indicate the best performance among all compared algorithms.

Table 17. Performance comparison of ASQS and competing algorithms with 10 fixed fog nodes and increasing IoT tasks: average latency, deadline satisfaction rate, load balance, runtime, and constraint violation.

Tasks	Metrics	MinMin	GA-PSO	QHS	ACO-PSO	RWHS	SJF	HACO	NF-MORL	EcoTaskSched	ASQS
50	AvgLatency	1.00	1.01	1.03	1.05	1.02	1.01	9.95×10^{-1}	1.00	1.00	9.89×10^{-1}
	DSR	9.79×10^{-1}	9.80×10^{-1}	8.59×10^{-1}	7.45×10^{-1}	9.45×10^{-1}	8.97×10^{-1}	9.97×10^{-1}	9.93×10^{-1}	9.89×10^{-1}	8.40×10^{-1}
	LoadBalance	1.10	1.43	1.70	3.00	1.55	1.39	9.30×10^{-1}	8.00×10^{-1}	9.90×10^{-1}	4.58×10^{-1}
	Runtime	6.33×10^{-1}	1.75×10^{-1}	3.95×10^{-1}	9.12×10^{-1}	2.39×10^{-1}	2.37×10^{-1}	2.49×10^{-1}	2.58×10^{-3}	7.74×10^{-3}	1.72×10^{-4}
	ConstraintViol	1.33×10^{-1}	1.82×10^{-1}	4.71	3.47×10^1	1.21	2.00	2.16×10^{-2}	4.57×10^{-2}	7.77×10^{-2}	1.09×10^1
100	AvgLatency	1.07	1.07	1.11	1.12	1.06	1.08	1.05	1.06	1.06	1.05
	DSR	5.45×10^{-1}	5.47×10^{-1}	5.30×10^{-1}	5.14×10^{-1}	5.38×10^{-1}	5.39×10^{-1}	5.57×10^{-1}	5.54×10^{-1}	5.52×10^{-1}	5.30×10^{-1}
	LoadBalance	1.33	1.31	2.62	4.81	1.92	1.62	1.39	8.80×10^{-1}	1.01	6.54×10^{-1}
	Runtime	6.51×10^{-1}	2.31×10^{-1}	5.66×10^{-1}	1.69	4.19×10^{-1}	3.06×10^{-1}	2.51×10^{-1}	4.30×10^{-3}	1.29×10^{-2}	2.87×10^{-4}
	ConstraintViol	1.09×10^2	1.13×10^2	1.66×10^2	2.49×10^2	1.32×10^2	1.38×10^2	9.57×10^1	9.83×10^1	1.02×10^2	1.52×10^2
200	AvgLatency	1.10	1.11	1.16	1.15	1.10	1.13	1.11	1.09	1.10	1.08
	DSR	2.55×10^{-1}	2.55×10^{-1}	2.45×10^{-1}	2.43×10^{-1}	2.49×10^{-1}	2.43×10^{-1}	2.45×10^{-1}	2.51×10^{-1}	2.52×10^{-1}	2.50×10^{-1}
	LoadBalance	2.01	2.38	5.08	7.04	3.07	2.92	2.80	1.30	1.63	7.22×10^{-1}
	Runtime	8.62×10^{-1}	3.27×10^{-1}	7.83×10^{-1}	3.29	1.05	4.31×10^{-1}	3.09×10^{-1}	1.04×10^{-2}	3.12×10^{-2}	6.94×10^{-4}
	ConstraintViol	1.22×10^3	1.24×10^3	1.40×10^3	1.57×10^3	1.30×10^3	1.32×10^3	1.30×10^3	1.25×10^3	1.26×10^3	1.30×10^3
400	AvgLatency	8.94×10^{-1}	8.95×10^{-1}	9.13×10^{-1}	9.88×10^{-1}	9.10×10^{-1}	9.11×10^{-1}	8.96×10^{-1}	8.90×10^{-1}	8.90×10^{-1}	8.86×10^{-1}
	DSR	1.51×10^{-1}	1.48×10^{-1}	1.44×10^{-1}	1.39×10^{-1}	1.48×10^{-1}	1.42×10^{-1}	1.46×10^{-1}	1.46×10^{-1}	1.47×10^{-1}	1.45×10^{-1}
	LoadBalance	2.40	2.67	5.91	3.00×10^1	5.15	4.27	3.19	1.31	1.73	5.82×10^{-1}
	Runtime	1.28	4.28×10^{-1}	1.56	6.80	5.27×10^{-1}	8.06×10^{-1}	6.98×10^{-1}	3.47×10^{-2}	1.04×10^{-1}	2.31×10^{-3}
	ConstraintViol	4.97×10^3	5.07×10^3	5.31×10^3	9.05×10^3	5.16×10^3	5.22×10^3	5.15×10^3	5.08×10^3	5.09×10^3	5.12×10^3
500	AvgLatency	9.59×10^{-1}	9.63×10^{-1}	1.01	1.01	9.57×10^{-1}	9.91×10^{-1}	1.00	9.50×10^{-1}	9.50×10^{-1}	9.43×10^{-1}
	DSR	1.24×10^{-1}	1.19×10^{-1}	1.19×10^{-1}	1.18×10^{-1}	1.18×10^{-1}	1.17×10^{-1}	1.16×10^{-1}	1.20×10^{-1}	1.20×10^{-1}	1.20×10^{-1}
	LoadBalance	2.88	4.24	1.12×10^1	2.07×10^1	8.05	6.59	9.78	2.08	2.74	9.13×10^{-1}
	Runtime	1.45	5.65×10^{-1}	2.01	8.76	1.81	1.01	9.16×10^{-1}	5.06×10^{-2}	1.52×10^{-1}	3.37×10^{-3}
	ConstraintViol	9.27×10^3	9.42×10^3	9.97×10^3	1.18×10^4	9.67×10^3	9.65×10^3	9.88×10^3	9.31×10^3	9.34×10^3	9.29×10^3

Note: Bold values indicate the best performance among all compared algorithms.

5.3.6. Statistical Significance Analysis on Small-Scale Experiments

To check whether the performance differences are statistically reliable, Friedman and Wilcoxon signed-rank tests were conducted based on the fitness results from 30 independent

runs. The Friedman test was used to compare all algorithms across the ten fog scheduling scenarios. The Wilcoxon signed-rank test was then used to compare ASQS with each baseline algorithm. Because several pairwise comparisons were performed, the Holm–Bonferroni correction was applied, and the significance level was set to 0.05.

As shown in Table 18, ASQS achieves the best average rank of 1.34 among all ten algorithms. The Friedman statistic is $\chi^2_F = 77.82$ with 9 degrees of freedom, and the corresponding p -value is smaller than 0.001. This result shows that the compared algorithms have statistically significant performance differences.

Table 18. Friedman test results over all fog scheduling scenarios.

Algorithm	Average Rank	Overall Ranking
ASQS	1.34	1
NF-MORL	2.51	2
EcoTaskSched	2.86	3
HACO	4.09	4
GA-PSO	4.73	5
ACO-PSO	5.18	6
RWHS	5.64	7
QHS	6.27	8
Min-Min	8.12	9
SJF	8.26	10

Note: Number of algorithms: 10; number of scenarios: 10; Friedman statistic: $\chi^2_F = 77.82$; degrees of freedom: 9; overall p -value: $p < 0.001$; significance level: 0.05; result: a significant difference exists among the compared algorithms.

Table 19 reports the corrected pairwise Wilcoxon results. All MFD values are positive, which means that ASQS obtains lower median fitness values than the baseline algorithms. In addition, all corrected p -values are below 0.05, indicating that the pairwise differences are statistically significant after multiple-comparison correction. The improvements over EcoTaskSched and NF-MORL are smaller than those over conventional baselines, but ASQS still obtains positive MFD values and more wins than losses.

These results show that ASQS has statistically reliable advantages across both node-scaling and task-scaling fog scheduling scenarios.

Table 19. Wilcoxon signed-rank test results with Holm–Bonferroni correction.

Compared Algorithm	MFD	W/T/L	Corrected p -Value	Result
SJF	0.2914	10/0/0	$p < 0.001$	Significant
Min-Min	0.2638	10/0/0	$p < 0.001$	Significant
QHS	0.1876	9/1/0	$p = 0.009$	Significant
RWHS	0.1542	8/2/0	$p = 0.018$	Significant
ACO-PSO	0.1669	8/1/1	$p = 0.022$	Significant
GA-PSO	0.1287	8/2/0	$p = 0.034$	Significant
HACO	0.1135	7/2/1	$p = 0.041$	Significant
EcoTaskSched	0.0712	7/3/0	$p = 0.048$	Significant
NF-MORL	0.0584	6/3/1	$p = 0.049$	Significant

Note: The reported p -values are corrected using the Holm–Bonferroni procedure. MFD: Median Fitness Difference. W/T/L denotes the number of scenarios where ASQS wins, ties, or loses against each baseline. The median fitness difference is computed as the baseline median fit minus that of ASQS. A positive value indicates that ASQS obtains a lower median fitness.

5.3.7. Large-Scale Task-Intensive Scalability Analysis

Table 20 and Figure 11 report the large-scale task-intensive scheduling results with 100 fog nodes. As the number of IoT tasks increases from 1000 to 8000, all algorithms show

higher scheduling costs and longer computational time due to the enlarged search space and heavier workload pressure. Nevertheless, ASQS maintains a stable and competitive performance trend across all scenarios.

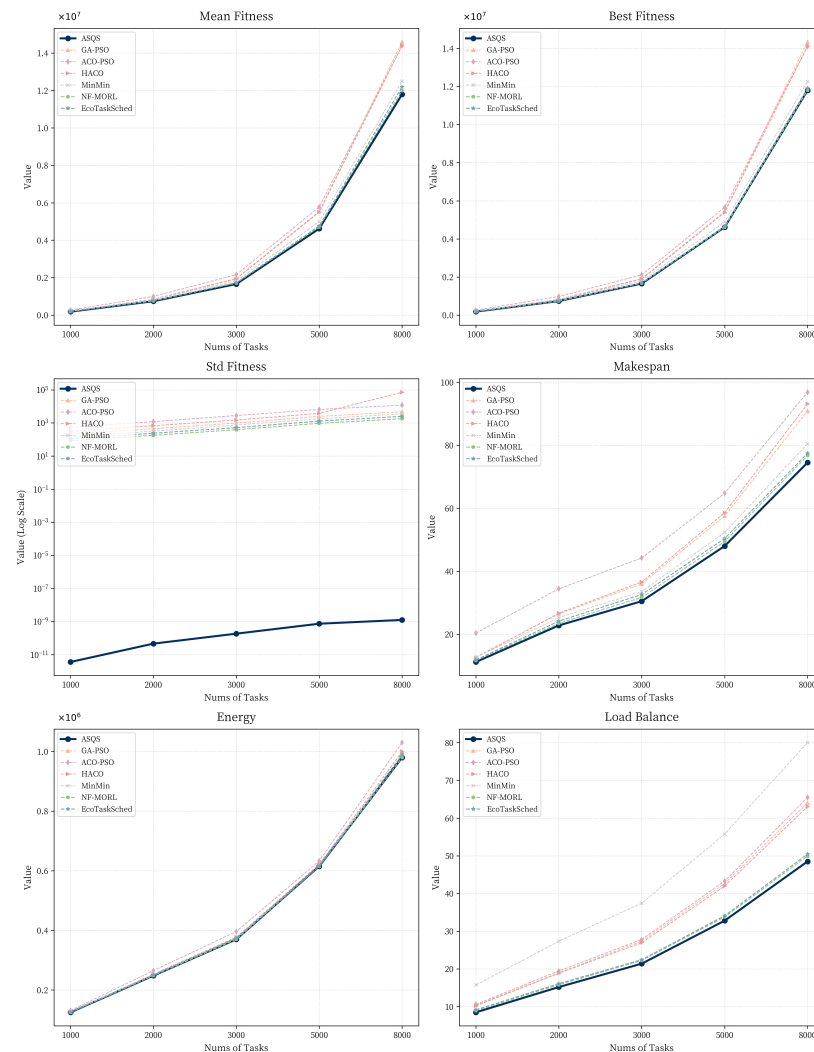


Figure 11. Performance trends of different algorithms under large-scale task-intensive IoT-Fog scheduling scenarios with 100 fog nodes.

ASQS achieves the lowest mean fitness in all five large-scale cases, indicating that its quality-source archive and multi-anchor synchronization mechanism remain effective under high-dimensional scheduling spaces. Moreover, ASQS obtains the best average latency and load-balance results in all scenarios, showing its ability to reduce response delay and avoid excessive workload concentration on a few fog nodes. For makespan, ASQS achieves the best results in most cases, although it does not dominate every scenario. This suggests that ASQS provides a balanced scheduling strategy rather than optimizing only one metric. In addition, ASQS maintains zero constraint violation across all large-scale scenarios, demonstrating strong feasibility preservation under intensive task loads.

Although ASQS requires more runtime than simple deterministic or lightweight methods, its computational cost is comparable to other population-based metaheuristics and is compensated by its superior scheduling quality. Overall, these results confirm that ASQS has good scalability and robustness in large-scale IoT-Fog task scheduling, especially when the task-node ratio increases substantially.

Table 20. Large-scale task-intensive fog scheduling results with 100 fog nodes.

Scenario	Metric	Min-Min	GA-PSO	ACO-PSO	HACO	EcoTaskSched	NF-MORL	ASQS
LS1	Mean Fitness	2.13×10^5	2.07×10^5	2.68×10^5	2.04×10^5	1.94×10^5	1.91×10^5	1.85×10^5
	Makespan	1.27×10^1	1.28×10^1	2.04×10^1	1.26×10^1	1.19×10^1	1.17×10^1	1.12×10^1
	AvgLatency	1.03	8.30×10^{-1}	7.80×10^{-1}	8.50×10^{-1}	7.30×10^{-1}	7.10×10^{-1}	6.60×10^{-1}
	LoadBalance	1.57×10^1	1.04×10^1	1.06×10^1	1.02×10^1	9.18	8.93	8.50
	Runtime	1.20×10^{-1}	1.25×10^1	1.58×10^1	1.42×10^1	8.20×10^{-1}	8.50×10^{-1}	1.35×10^1
	ConstraintViol.	1.80×10^1	2.00	4.00	4.00	0.00	0.00	0.00
LS2	Mean Fitness	8.29×10^5	8.51×10^5	9.99×10^5	8.44×10^5	7.70×10^5	7.55×10^5	7.40×10^5
	Makespan	2.53×10^1	2.65×10^1	3.44×10^1	2.67×10^1	2.42×10^1	2.35×10^1	2.81×10^1
	AvgLatency	1.77	1.38	1.39	1.47	1.27	1.24	1.23
	LoadBalance	2.74×10^1	1.90×10^1	1.95×10^1	1.89×10^1	1.61×10^1	1.58×10^1	1.52×10^1
	Runtime	2.50×10^{-1}	2.48×10^1	3.12×10^1	2.85×10^1	1.60	1.65	2.68×10^1
	ConstraintViol.	3.40×10^1	3.00	3.00	5.00	0.00	0.00	0.00
LS3	Mean Fitness	1.83×10^6	1.96×10^6	2.16×10^6	1.93×10^6	1.73×10^6	1.71×10^6	1.66×10^6
	Makespan	3.36×10^1	3.60×10^1	4.42×10^1	3.66×10^1	3.26×10^1	3.17×10^1	3.05×10^1
	AvgLatency	2.42	2.01	1.93	2.08	1.83	1.82	1.69
	LoadBalance	3.75×10^1	2.74×10^1	2.78×10^1	2.70×10^1	2.25×10^1	2.23×10^1	2.14×10^1
	Runtime	3.80×10^{-1}	3.82×10^1	4.85×10^1	4.41×10^1	2.38	2.45	4.15×10^1
	ConstraintViol.	6.30×10^1	1.10×10^1	1.20×10^1	1.70×10^1	0.00	0.00	0.00
LS4	Mean Fitness	4.99×10^6	5.54×10^6	5.78×10^6	5.50×10^6	4.76×10^6	4.71×10^6	4.62×10^6
	Makespan	5.23×10^1	5.76×10^1	6.48×10^1	5.86×10^1	5.04×10^1	4.95×10^1	4.80×10^1
	AvgLatency	3.78	3.04	2.96	3.28	2.97	2.84	2.82
	LoadBalance	5.58×10^1	4.26×10^1	4.33×10^1	4.20×10^1	3.41×10^1	3.38×10^1	3.28×10^1
	Runtime	6.50×10^{-1}	6.54×10^1	8.21×10^1	7.52×10^1	4.05	4.15	7.08×10^1
	ConstraintViol.	8.50×10^1	2.30×10^1	6.00	7.00	0.00	1.00	0.00
LS5	Mean Fitness	1.25×10^7	1.46×10^7	1.44×10^7	1.44×10^7	1.22×10^7	1.20×10^7	1.18×10^7
	Makespan	8.05×10^1	9.09×10^1	9.69×10^1	9.31×10^1	7.75×10^1	7.67×10^1	7.45×10^1
	AvgLatency	5.46	4.60	4.32	4.60	4.23	4.40	4.17
	LoadBalance	8.00×10^1	6.40×10^1	6.55×10^1	6.31×10^1	5.04×10^1	5.00×10^1	4.85×10^1
	Runtime	1.05	1.09×10^2	1.35×10^2	1.25×10^2	6.70	6.85	1.18×10^2
	ConstraintViol.	1.27×10^2	2.80×10^1	7.00	1.70×10^1	0.00	1.00	0.00

Note: Bold values indicate the best performance among all compared algorithms.

6. Conclusions and Future Work

This paper proposed ASQS, a quality-source-driven multi-anchor synchronized search algorithm for IoT-Fog task scheduling. By extracting representative quality-source anchors from layered populations and reusing them through an ACO-inspired probabilistic synchronization mechanism, ASQS improves the utilization of historical high-quality search information and reduces the dependence on a single elite solution.

Extensive experiments, including benchmark-function tests, ablation studies, sensitivity analysis, IoT-Fog scheduling experiments, large-scale task-intensive scenarios, and statistical significance tests, were conducted to evaluate the proposed algorithm. The results show that ASQS achieves competitive optimization accuracy and convergence stability on benchmark functions, while the ablation study verifies the contribution of its key components. In fog scheduling scenarios, ASQS obtains strong comprehensive scheduling performance in terms of fitness, makespan, latency, load balance, and constraint handling. In particular, the large-scale experiment with 100 fog nodes and up to 8000 IoT tasks demonstrates that ASQS can maintain stable performance under increased task dimensions and heavier workload pressure. These findings indicate that ASQS is an effective, robust, and scalable optimizer for resource-constrained IoT-Fog task scheduling.

Despite these results, several limitations remain. The current model focuses on static batch scheduling and does not fully consider dynamic task arrivals, node failures, bandwidth fluctuations, workload changes, or online task migration. In addition, the objective weights and penalty coefficients are predefined, and the algorithm still involves several control parameters. Moreover, although simulation-based experiments provide reproducible evaluation, further validation on real workload traces and fog/edge platforms is required.

Future work will extend ASQS toward dynamic and online computational scheduling in IoT–Fog environments. This includes handling continuously arriving tasks, time-varying resource states, adaptive task migration, and dynamic resource allocation. Adaptive weight adjustment, self-adaptive parameter control, and Pareto-based multi-objective optimization will also be explored to improve flexibility under diverse QoS requirements. Finally, real-world traces and practical fog/edge testbeds will be used to further evaluate the scalability and deployment feasibility of ASQS in realistic dynamic computing environments.

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

Funding: This work was supported by the National Natural Science Foundation of China (62376089, U23A20318, 62302154), the Natural Science Foundation of Hubei Province (2024AFB882), and the Program for Scientific and Technological Innovation Teams of Young and Middle Aged Researchers in Higher Education Institutions of Hubei Province (T2023007).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Xianjing Zhou was employed by Wuhan Zhuoer Information Technology Co., Ltd. Authors Donglei Xu and Mingming Zhao were employed by Wuhan Fiberhome Technical Services Co., Ltd. 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. Mindil, A.; Hamed, A.Y.; Hassan, M.R.; Elnahary, M.K. A novel approach for dynamic task scheduling for IoT in fog-cloud environment. *Sci. Rep.* **2026**, *16*, 5501. [\[CrossRef\]](#)
2. Khan, A.; Ullah, F.; Shah, D.; Khan, M.; Ali Aurakzai, E.S.; Tahir, M. EcoTaskSched: A hybrid machine learning approach for energy-efficient task scheduling in IoT-based fog-cloud environments. *Sci. Rep.* **2025**, *15*, 12296. [\[CrossRef\]](#)
3. Heidarpour, A.R.; Heidarpour, M.R.; Ardakani, M.; Tellambura, C.; Uysal, M. DeepWFFS: Enhancing fog computing efficiency through multiqueue architecture and intelligent controller for task prioritization. *IEEE Internet Things J.* **2024**, *11*, 27030–27042. [\[CrossRef\]](#)
4. Chuan, W.C.; Laghari, S.U.A.; Manickam, S.; Ashraf, E.; Karuppayah, S. Challenges and opportunities in fog computing scheduling: A literature review. *IEEE Access* **2025**, *13*, 14702–14726. [\[CrossRef\]](#)
5. Hosseini, S.M.; Shirvani, M.H.; Motameni, H. Multi-objective discrete Cuckoo search algorithm for optimization of bag-of-tasks scheduling in fog computing environment. *Comput. Electr. Eng.* **2024**, *119*, 109480. [\[CrossRef\]](#)
6. Pan, S.; Huang, C.; Fan, J.; Shi, Z.; Tong, J.; Wang, H. Internet of Things fog computing: Through Lyapunov-based long short-term memory particle swarm optimization algorithm for energy consumption optimization. *Sensors* **2024**, *24*, 1165. [\[CrossRef\]](#)
7. Abawajy, J.; Ghanavati, S.; Izadi, D. Enhancing dependability of fog computing using learning-based task scheduling. In *IEEE Transactions on Dependable and Secure Computing*; IEEE: Piscataway, NJ, USA, 2025.
8. Alsadie, D.; Alsulami, M. Modified grey wolf optimization for energy-efficient internet of things task scheduling in fog computing. *Sci. Rep.* **2025**, *15*, 14730. [\[CrossRef\]](#)
9. Bhakhar, R.; Chhillar, R.S. Dynamic multi-criteria scheduling algorithm for smart home tasks in fog-cloud IoT systems. *Sci. Rep.* **2024**, *14*, 29957. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Choppara, P.; Lokesh, B. Efficient task scheduling and load balancing in fog computing for crucial healthcare through deep reinforcement learning. *IEEE Access* **2025**, *13*, 26542–26563. [\[CrossRef\]](#)
11. Moazzami, S.; Mirzaei, A.; Aminian, M.; Karimi, R.; Mikaeilvand, N. A Hybrid Fuzzy Logic and Deep Reinforcement Learning Algorithm for Adaptive Task Scheduling and Resource Allocation in Heterogeneous Fog–Cloud Environments. *Sustain. Comput. Inform. Syst.* **2025**, *50*, 101260. [\[CrossRef\]](#)

12. Heirati, A.; Fartash, M.; Khalily-Dermany, M. Optimized task scheduling in fog-cloud computing using hybrid deep learning and metaheuristic algorithms. *Neural Process. Lett.* **2025**, *58*, 10. [\[CrossRef\]](#)
13. Yu, X.; Tang, L.; Mi, J.; Long, L.; Qin, X.; Li, X.; Mo, Q. NF-MORL: A neuro-fuzzy multi-objective reinforcement learning framework for task scheduling in fog computing environments. *Sci. Rep.* **2025**, *16*, 2455. [\[CrossRef\]](#) [\[PubMed\]](#)
14. Ali, S.; Alubady, R. A Survey of Fog Computing-Based Resource Allocation Approaches: Overview, Classification, and Opportunity for Research. *Iraqi J. Sci.* **2024**, *65*, 4008–4029. [\[CrossRef\]](#)
15. Alshammari, N.; Gill, S.S.; Pervaiz, H.; Ni, Q.; Ahmed, H. Resource Scheduling in Integrated IoT and Fog Computing Environments: A Taxonomy, Survey and Future Directions. In *Resource Management in Distributed Systems*; Springer Nature: Singapore, 2024; pp. 63–77.
16. Isong, B.; Mamidza, F. Computational Task Scheduling Across IoT-Edge-Fog-Cloud Continua: Algorithms, Adaptability and Research Gaps. *Indones. J. Comput. Sci.* **2026**, *15*, 1–15. [\[CrossRef\]](#)
17. Aldossary, D.; Aldahasi, E.; Balharith, T.; Helmy, T. A Systematic Literature Review on Load-Balancing Techniques in Fog Computing: Architectures, Strategies, and Emerging Trends. *Computers* **2025**, *14*, 217. [\[CrossRef\]](#)
18. Aminu, J.; Latip, R.; Hanafi, Z.M.; Kamarudin, S.; Gabi, D. Systematic review of metaheuristic-based task scheduling strategies in edge computing environments. *Discov. Comput.* **2025**, *28*, 307. [\[CrossRef\]](#)
19. Zhao, L.; Lin, Z.; Shao, H.; Zheng, W.; Zhang, S.; Wang, Y.; Li, Q. Improved task scheduling and load balancing in fog computing infrastructure using hybrid evolutionary method. *Clust. Comput.* **2025**, *28*, 396. [\[CrossRef\]](#)
20. Abraham, O.L.; Ngadi, M.A.B.; Sharif, J.B.M.; Mohd Sidik, M.K. Multi-objective optimization techniques in cloud task scheduling: A systematic literature review. *IEEE Access* **2025**, *13*, 12255–12291. [\[CrossRef\]](#)
21. Ali, E.M.; Abawajy, J.; Lemma, F.; Baho, S.A. Analysis of deep reinforcement learning algorithms for task offloading and resource allocation in fog computing environments. *Sensors* **2025**, *25*, 5286. [\[CrossRef\]](#) [\[PubMed\]](#)
22. Nemati, A.M.; Mansouri, N. Resource allocation in fog computing: A survey on current state and research challenges. *Knowl. Inf. Syst.* **2025**, *67*, 2091–2170. [\[CrossRef\]](#)
23. Momeny, S.; Robatmily, M.; Rahmani, A.M. Multi-objective optimization for the performance of task scheduling in homogeneous fog networks. *Clust. Comput.* **2025**, *28*, 796. [\[CrossRef\]](#)
24. Vali, A.A.; Azizi, S.; Shojafar, M.; Buyya, R. Energy-Efficient Resource Management in Microservices-Based Fog and Edge Computing: State-of-the-Art and Future Directions. *ACM Comput. Surv.* **2026**, *58*, 1–42. [\[CrossRef\]](#)
25. Ghafari, R.; Mansouri, N. Swarm intelligence techniques and their applications in fog/edge computing: An in-depth review. *Artif. Intell. Rev.* **2025**, *58*, 359. [\[CrossRef\]](#)
26. Thakur, R.; Sikka, G.; Bansal, U.; Giri, J.; Mallik, S. Deadline-aware and energy efficient IoT task scheduling using fuzzy logic in fog computing. *Multimed. Tools Appl.* **2025**, *84*, 14359–14386. [\[CrossRef\]](#)
27. Long, G.; Wang, S.; Lv, C. QoS-aware resource management in cloud computing based on fuzzy meta-heuristic method. *Clust. Comput.* **2025**, *28*, 276. [\[CrossRef\]](#)
28. Abdulwahab, S.; Askar, M.I.S.; Hussien, D. Comprehensive review of advanced machine learning strategies for resource allocation in fog computing systems. *Indones. J. Comput. Sci.* **2025**, *14*. [\[CrossRef\]](#)
29. Nasiri Aghdam, Z.; Rezaee, A.; Rahmani, A.M.; Hosseinzadeh, M. Evolutionary-based hybrid classical-fuzzy heuristic protocol for real-time data availability of IoHT devices in cloud-fog computing. *Clust. Comput.* **2025**, *28*, 936. [\[CrossRef\]](#)
30. Cai, T.; Zhang, S.; Ye, Z.; Zhou, W.; Wang, M.; He, Q.; Chen, Z.; Bai, W. Cooperative metaheuristic algorithm for global optimization and engineering problems inspired by heterosis theory. *Sci. Rep.* **2024**, *14*, 28876. [\[CrossRef\]](#)
31. Ye, Z.; Ma, L.; Chen, H. A hybrid rice optimization algorithm. In *Proceedings of the 2016 11th International Conference on Computer Science & Education (ICCSE)*; IEEE: Piscataway, NJ, USA, 2016; pp. 169–174.
32. Taheri, A.; RahimiZadeh, K.; Baumbach, J.; Beheshti, A.; Zolotareva, O.; Al-Betar, M.A.; Mirjalili, S.; Gandomi, A.H. Farthest better or nearest worse optimizer: A novel metaheuristic algorithm. *Artif. Intell. Rev.* **2026**, *59*, 79. [\[CrossRef\]](#)
33. Liao, T.; Socha, K.; de Oca, M.A.M.; Stützle, T.; Dorigo, M. Ant colony optimization for mixed-variable optimization problems. *IEEE Trans. Evol. Comput.* **2013**, *18*, 503–518. [\[CrossRef\]](#)
34. Qin, F.; Zain, A.M.; Zhou, K.Q.; Norfadzlan, Y.; Dwi Prasetya, D.; Rozita, A.J.; Zaheera, Z.A.; Mahadi, B.; Yusri, K.; Mazlina, A.M. Hybrid harmony search algorithm integrating differential evolution and lévy flight for engineering optimization. *IEEE Access* **2025**, *13*, 13534–13572. [\[CrossRef\]](#)
35. Bujok, P.; Lacko, M.; Kolenovsky, P. Differential evolution and engineering problems. *Mendel* **2023**, *29*, 45–54. [\[CrossRef\]](#)
36. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
37. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [\[CrossRef\]](#)
38. Chen, M.; Feng, C.; Cheng, R. MetaDE: Evolving differential evolution by differential evolution. *IEEE Trans. Evol. Comput.* **2025**, *30*, 141–155. [\[CrossRef\]](#)
39. Zhang, J.; Huang, Y.; Wang, Y.; Ma, G. Multi-objective optimization of concrete mixture proportions using machine learning and metaheuristic algorithms. *Constr. Build. Mater.* **2020**, *253*, 119208. [\[CrossRef\]](#)

40. Zhou, X.; Gui, W.; Heidari, A.A.; Cai, Z.; Liang, G.; Chen, H. Random following ant colony optimization: Continuous and binary variants for global optimization and feature selection. *Appl. Soft Comput.* **2023**, *144*, 110513. [[CrossRef](#)]
41. Khosla, T.; Verma, O. An adaptive hybrid particle swarm optimizer for constrained optimization problem. In Proceedings of the 2021 International Conference in Advances in Power, Signal, and Information Technology (APSIT 2021), Dehradun, India, 8–10 October 2021; pp. 134–139.
42. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
43. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; MIT Press: Cambridge, MA, USA, 1992.
44. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A new heuristic optimization algorithm: Harmony search. *Simulation* **2001**, *76*, 60–68. [[CrossRef](#)]
45. Dorigo, M.; Maniezzo, V.; Colnori, A. Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1996**, *26*, 29–41. [[CrossRef](#)]
46. Silberschatz, A.; Galvin, P.B.; Gagne, G. *Operating System Concepts*, 10th ed.; Wiley: Hoboken, NJ, USA, 2018.
47. Tawfeek, M.A.; El-Sisi, A.; Keshk, A.E.; Torkey, F.A. Cloud task scheduling based on ant colony optimization. In Proceedings of the 8th International Conference on Computer Engineering and Systems, Cairo, Egypt, 26–28 November 2013; pp. 64–69.
48. Maheswaran, M.; Ali, S.; Siegel, H.J.; Hensgen, D.; Freund, R.F. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In Proceedings of the 8th Heterogeneous Computing Workshop, San Juan, Puerto Rico, 12 April 1999; pp. 30–44.

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.