Global Journal of Computer Science and Technology - Section B
GJCST

LJWOA: A Lévy-Flight-Enhanced Jaya–Whale Hybrid for Latency- and Power-Aware Microservice Placement in Fog-Cloud Environments

Nikhil Anand*, Rohan Kukreti, and Prince Kumar Raj

Abstract

Placing containerized microservices across a fog-cloud continuum requires jointly optimizing network latency, node power consumption, and load balance. This paper studies LJWOA, a three-stage hybrid metaheuristic that adds a Jayastyle elitist refinement and a Lévy-flight restart of the weakest individuals to the Whale Optimization Algorithm (WOA). Evaluation uses six benchmark functions (dimension 30) and synthetic fog-cloud placement instances with 50–300 microservices (CPU workload 200–1500 MI, payload 1–50 MB) on 12–36 heterogeneous fog and cloud nodes, with 15–30 independent runs per setting and Holm-corrected Wilcoxon signed-rank tests. In the original experiments LJWOA attains a lower weighted objective than Jaya, GWO, and PSO at all five scales and than WOA at 50 and 100 microservices, is statistically indistinguishable from WOA at 150–200, and is significantly worse at 300. Additional experiments requested during review use an equal function-evaluation budget and an independent re-implementation: LJWOA remains ahead of Jaya, GWO, SMA, and NSGA-II at every scale and ties with PSO, but its advantage over WOA is not significant at 50–200 and reverses at 300 (0.612 vs. 0.576), and GA and L-SHADE reach lower objective values at all scales. The Lévy stage improves on the twostage variant on held-out instances (raw $p < 0.01$ at every scale), whereas the stage order has no consistent effect. LJWOA costs 1.36–1.43× WOA per iteration, and all algorithms overload on average 53–91% of nodes under the soft capacity constraint. LJWOA is therefore a modest improvement within the WOA family rather than a state-of-the-art placement method.

Keywords: *Cloud Computing, container scheduling, fog computing, Jaya algorithm, Lévy flight, metaheuristic optimization, microservice placement, power-aware computing, Whale Optimization Algorithm*

* **Corresponding author** Nikhil Anand

1. Introduction

The shift from monolithic applications to containerized microservices, combined with the growth of latency-sensitive Internet of Things (IoT) workloads, has made the fog-cloud continuum an important deployment setting for modern distributed systems. Fog nodes sit close to end users and offer low communication latency but limited compute capacity, while cloud nodes offer abundant capacity at the cost of higher latency and, typically, higher power draw. Deciding which fog or cloud node should host each microservice (the placement problem) is an NP-hard combinatorial optimization problem: an exhaustive search is intractable even for modest numbers of microservices and nodes, which is why the literature has increasingly turned to metaheuristic optimization.

Prior work in this space has applied ant colony optimization (ACO), the Grey Wolf Optimizer (GWO), and the Jaya algorithm, individually and in hybrid form, to communication-aware virtual machine (VM) placement and fog device deployment [1, 2, 3, 4, 5]. These studies show that hybridizing two metaheuristics can combine complementary exploration and exploitation behaviours to outperform either algorithm alone. Container/microservice placement has received substantially more attention in recent work, including dependency-aware, learning-based, and multi-objective approaches [18, 19, 20, 21, 23]. What remains open is whether a lightweight hybrid combination, coupling Jaya's parameter-free exploitation, WOA's population-wide exploratory search, and an explicit diversification mechanism, can achieve a statistically validated solution-quality advantage rather than only a speed/quality trade-off, and how such a hybrid compares with stronger evolutionary baselines under an equal evaluation budget.

An earlier, two-stage version of this hybrid (WOA exploration plus Jaya-style elitist refinement, without a diversification stage) matched WOA's solution quality and converged faster, but did not exceed WOA's quality on the placement problem. This paper adds a third stage (an unconditional Lévy-flight-based restart of the weakest individuals each iteration) and examines whether this closes the quality gap with WOA. In the original experiments it does at 50 and 100 microservices; in the additional equal-budget experiments of Sections 6.5–6.12 the two algorithms are comparable at 50–200 microservices and WOA is better at 300. The third stage also costs slower convergence and higher per-iteration compute. We report this development process, including the design that did not work as well, because we consider the negative result (Section 4.2) informative for readers considering similar hybridizations.

This paper makes the following contributions: (1) we formulate a latency- and power-aware microservice placement problem for fog-cloud environments as a continuous optimization problem, combining processing time, node power, payload-weighted network latency, and load-balance terms into a weighted objective with a soft node-capacity constraint; (2) we propose LJWOA, a three-stage hybrid: population-wide WOA exploration, Jaya-style elitist exploitation, and an unconditional Lévy-flight restart that is intended to inject diversity by replacing the weakest individuals each iteration regardless of whether the replacement immediately improves on what it replaces (Section 6.12 shows that it does not prevent early loss of population diversity); (3) we show that this third stage is landscape-dependent: it nominally improves Sphere (does not survive correction) and significantly degrades Rosenbrock and Schwefel on the benchmark suite, while improving the placement

objective at the scales examined in the restart-fraction ablation; we therefore evaluate two configurations of the same algorithm rather than force one to fit both; (4) we apply explicit global-best bookkeeping uniformly to all five compared algorithms, so no algorithm's result can regress below an intermediate solution it already found; and (5) we validate LJWOA against Jaya, WOA, GWO, and PSO using 15–20 independent runs per configuration, paired Wilcoxon signed-rank tests, and a restart-fraction ablation, and, in response to review, extend the comparison to equal-budget runs against HHO, SMA, L-SHADE, GA, and NSGA-II, a held-out tuning of $D_{restart}$, a stage-order ablation, capacity-violation and weight-sensitivity analyses, and a MILP reference, reporting the full result, including the case (300 microservices) where LJWOA is significantly behind WOA and the stronger GA and L-SHADE baselines.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 formulates the placement problem. Section 4 presents LJWOA. Section 5 describes the experimental setup. Section 6 reports results, including the additional experiments of Sections 6.5–6.12. Section 7 concludes.

2. Related Work

Service and VM placement in fog-cloud systems has been addressed through heuristic, metaheuristic, and learning-based techniques; a recent systematic review highlights multi-objective latency-power-utilization trade-offs as a persistent, open challenge [6]. Container-based placement has received targeted attention: a two-tier multi-objective scheme for container-based fog-cloud platforms [7], ant-colony scheduling with container migration for DAG tasks [8], nature-inspired fog node placement [9], and Harris Hawks-based multi-objective service placement [10]. More recent work broadens this further: federated fog placement [19]; dependency- and runtime-aware multi-tier placement [20]; deep-RL-based hybrid fog/cloud scheduling [21]; multi-objective cost/makespan CaaS placement [22]; and swarm-optimized service-function-chain deployment [23]. These studies reinforce that placement research is increasingly concerned with dependencies, resource constraints, and deployment-level objectives rather than node assignment alone.

Our prior work has focused on communication-aware VM and fog-device placement using ACO, GWO, and Jaya, both individually and in hybrid combinations [1, 2, 3, 4, 5].

These studies motivate the use of hybrid metaheuristic search but do not address the specific WOA–Jaya–Lévy sequence investigated here. The base algorithms considered are Jaya [11], WOA [12], GWO [13], PSO [14], and Lévy flight [15]. The individual components of LJWOA are not themselves new: Jaya has previously been combined with a Lévy-flight operator for energy-aware fog module placement [16], while Lévy-flight-enhanced WOA variants have been proposed for global optimization and scheduling [17]. The contribution of LJWOA is therefore not the introduction of these individual operators, but their specific three-stage sequential integration: WOA exploration, Jaya-style elitist exploitation, and unconditional Lévy perturbation of only the weakest individuals. The additional baselines used in the revised evaluation are Harris Hawks Optimization (HHO) [24], the Slime Mould Algorithm (SMA) [25], L-SHADE with linear population-size reduction [26], and the multi-objective genetic algorithm NSGA-II [27]; HHO has previously been used for multi-objective fog service placement [10]. The proposed architecture is evaluated through statistical comparisons and ablation experiments to characterize when the Lévy stage helps or harms, and is subsequently applied to latency- and power-aware fog-cloud microservice placement.

3. Problem Formulation

We consider a fog-cloud system with nodes $N = \{1, \dots, n\}$ comprising n_{fog} fog nodes and n_{cloud} cloud nodes, and microservices $M = \{1, \dots, m\}$ to be placed. Each microservice i has a CPU workload w_i (MI) and payload size d_i (MB). Each node k has processing capacity C_k (MIPS), link latency ℓ_k (ms/MB), static power P_k^{stat} , and dynamic power coefficient P_k^{dyn} . A placement is a mapping $x : M \to N$. Given x, let $L_k = \sum_{i:x(i)=k} w_i$ and $u_k = L_k/C_k$. Capacity should nominally satisfy $L_k \le C_k$; this is enforced softly (the makespan term grows with u_k for $u_k > 1$), consistent with soft-constraint handling in prior continuous-encoding placement work [1, 2, 3, 4, 5].

Four objective terms are defined: makespan (bottleneck processing time),

$$T(x) = \max_k (L_k/C_k); \tag{1}$$

power over active nodes,

$$P(x) = \sum_{k:L_k>0} \left[P_k^{stat} + P_k^{dyn} \cdot \min(u_k, 1)\right]; \tag{2}$$

payload-weighted communication cost,

$$G(x) = \sum_i d_i \cdot \ell_{x(i)}; \tag{3}$$

and load imbalance,

$$B(x) = \mathrm{std}_k(u_k). \tag{4}$$

Because $\min(u_k, 1)$ caps the dynamic power of an overloaded node, overload is penalised only through the makespan term (1); capacity violations are therefore reported explicitly in Section 6.9. $G(x)$ is a node-tier access-latency proxy, not a full inter-microservice dependency model; a traffic-graph-aware extension is future work. Each term is normalized by a fixed reference (T_{ref} : single fastest node; P_{ref} : full-system power at full utilization; G_{ref} : routing every payload through the highest-latency node) and combined into a weighted objective

$$F(x) = w_1\hat{T}(x) + w_2\hat{P}(x) + w_3\hat{G}(x) + w_4B(x), \tag{5}$$

with $w_1 = w_2 = 0.35$, $w_3 = 0.20$, $w_4 = 0.10$, weighting makespan and power as primary concerns, communication cost secondary, and load balance a regularizer, consistent with our prior placement studies [1, 2, 3, 4, 5].

The underlying problem is discrete; we use a continuous surrogate encoding $x \in [0, n)^m$ so real-valued metaheuristics apply directly. Every optimizer clips each component to $[0, n-\epsilon]$ before evaluation, and node assignment is obtained by $x(i) = \lfloor x_i \rfloor$, guaranteeing every evaluated candidate maps to a valid node.

4. Proposed LJWOA Algorithm

4.1. Background: Jaya, woa, and Lévy Flight

Jaya is parameter-free: each candidate moves toward the best and away from the worst solution in the population [11]. With X_{best} , X_{worst} and $r_1, r_2 \sim U(0, 1)$ per dimension,

$$X_i' = X_i + r_1 \cdot (X_{best} - |X_i|) - r_2 \cdot (X_{worst} - |X_i|). \tag{6}$$

PREPRINT

WOA models humpback-whale bubble-net hunting [12]. With a shrinking linearly from 2 to 0, $A = 2ar_1 - a$, $C = 2r_2$, and $p \sim U(0,1)$: for p < 0.5, $|A| < 1$ (encircling),

$$X_i(t+1) = X^*(t) - A \cdot D, \quad D = |C \cdot X^*(t) - X_i(t)|; \qquad (7)$$

for $p < 0.5$, $|A| \geq 1$, X^* is replaced by a random member (exploration); for $p \geq 0.5$ (spiral attack), with $b = 1$ and $l \sim U(-1,1)$,

$$X_i(t+1) = D' \cdot e^{bl} \cos(2\pi l) + X^*(t), \quad D' = |X^*(t) - X_i(t)|. \quad (8)$$

Lévy steps are drawn via Mantegna's algorithm [28] with $\beta = 1.5$: draw $u \sim \mathcal{N}(0, \sigma_u^2)$ (with σ_u from the standard Mantegna formula) and $v \sim \mathcal{N}(0,1)$, and form

$$L(\beta) = u/|v|^{1/\beta}. \qquad (9)$$

$L(\beta)$ is heavy-tailed: most steps are small, but occasional large steps let a candidate reach distant, unexplored regions.

4.2. A Two-Stage Precursor

Our first design combined only WOA exploration (Stage 1) with Jaya-style elitist refinement of the top fraction of the population (Stage 2), both under greedy selection. This design (JWOA) matched WOA's quality on most benchmark functions and placement scales and converged faster at 100–300 microservices, but was significantly worse than WOA at 300 microservices and tied elsewhere, without a genuine quality advantage.

4.3. LJWOA: Adding Lévy-Flight Restart

Because Jaya's and WOA's operators use greedy acceptance, once the population converges onto a shared region neither has an explicit mechanism to leave it. LJWOA adds a third stage: after Stages 1–2, the worst $D_{restart}$ fraction of the population is unconditionally replaced with Lévy-flight jumps anchored at the current best solution, with the aim of injecting diversity the greedy stages alone cannot provide (Section 6.12 examines whether this is achieved). This is not universally beneficial: it degrades the Rosenbrock and Schwefel benchmark functions while improving placement quality at 150 and 300 microservices (the two scales examined in the restart-fraction ablation). We therefore treat $D_{restart}$ as domain-tunable: $D_{restart} = 0$ (Stage 3 disabled, reducing to two-stage JWOA) for generic benchmark validation, and $D_{restart} = 0.1$ for the placement application; Section 6.7 re-examines this choice on separate tuning instances.

4.4. Algorithm

Let P be the population size, $D_{elite} = 0.3$ the Jaya-refinement elite fraction, $D_{restart} \in \{0, 0.1\}$ the Lévy-restart fraction, and I_{max} the iteration budget.

Per-iteration cost is $O((1+D_{elite}+D_{restart})P)$ evaluations, roughly 1.4× plain WOA at placement settings ($D_{elite} = 0.3$, $D_{restart} = 0.1$), matching the measured wall-clock overhead (Section 6.3). To keep the comparison fair, we additionally apply the same global-best bookkeeping to WOA and GWO, comparing each iteration's best to the incumbent and replacing only on improvement, without altering their core update rules, since PSO and Jaya already track a best-so-far solution natively.

Algorithm 1 LJWOA (Lévy-Jaya-Whale Optimization Algorithm)

1: Initialize population $X_1, \ldots, X_P$ uniformly at random
2: **for** $t = 1$ to I_{max} **do**
3: $a \leftarrow 2 - 2t/I_{max}$
4: **Stage 1:** apply WOA update (7)–(8) to every X_i; clip, evaluate, greedily accept
5: **Stage 2, if** $D_{elite} > 0$**:** re-rank; apply Jaya update (6) to top $[D_{elite}P]$; clip, evaluate, greedily accept
6: **Stage 3, if** $D_{restart} > 0$**:** unconditionally replace worst $[D_{restart}P]$ with X_{best} + scale · $L(\beta)$, scale = $0.03(ub - lb)a$
7: Update best-so-far
8: **end for**
9: **return** best solution found

5. Experimental Setup

JWOA (LJWOA with $D_{restart} = 0$) is validated against Jaya, WOA, GWO, and PSO on six standard benchmark functions (Table 1) spanning unimodal (F1–F2) and multimodal (F3–F6) landscapes, at dimension 30, population 30, 200 iterations, 20 runs per pair. LJWOA ($D_{restart} = 0.1$) is evaluated on five synthetic fog-cloud scenarios of increasing scale (Table 2), with parameter ranges chosen to be broadly representative of heterogeneous deployments: microservice CPU workload ∼ $U[200, 1500]$ MI, payload ∼ $U[1, 50]$ MB; fog capacity ∼ $U[800, 2500]$ MIPS, latency ∼ $U[0.5, 2.0]$ ms/MB, power ∼ $U[30, 60]/[40, 80]$ W; cloud capacity ∼ $U[4000, 10000]$ MIPS, latency ∼ $U[4.0, 9.0]$ ms/MB, power ∼ $U[100, 160]/[150, 250]$ W. All algorithms use population 30, 150 iterations, 15 runs per (scenario, algorithm) pair. Table 3 lists each algorithm's configuration.

Function	Type	Range	Optimum
F1 Sphere	Unimodal	$[-100, 100]$	0
F2 Rosenbrock	Unimodal, narrow valley	$[-30, 30]$	0
F3 Rastrigin	Multimodal	$[-5.12, 5.12]$	0
F4 Ackley	Multimodal	$[-32, 32]$	0
F5 Griewank	Multimodal	$[-600, 600]$	0
F6 Schwefel	Multimodal, deceptive	$[-500, 500]$	0

Table 1. Benchmark functions used for validation

Microservices	Fog nodes	Cloud nodes	Total nodes
50	8	4	12
100	12	6	18
150	16	8	24
200	20	10	30
300	24	12	36

Table 2. Fog-cloud placement scenario configurations

For each test instance (6 benchmarks + 5 placement scales = 11), LJWOA's per-run final fitness is compared against each baseline using the two-sided Wilcoxon signed-rank test (p < 0.05), with Holm-Bonferroni correction applied within each family (24 benchmark tests, 20 placement tests). Tests are paired by run index: for a given (instance, run) pair every algorithm shares the same random seed, producing an identical initial population and, for placement, an identical problem instance across all five algorithms. We also compute each algorithm's average rank (1 = best) across all 11 instances as a descriptive summary, not a formal Friedman test. To validate $D_{restart} = 0.1$ as a considered choice, we repeat the placement experiment at 150 and 300 microservices with $D_{restart} \in \{0, 0.05, 0.1, 0.15, 0.2, 0.3\}$.

EARLY VIEW

Algorithm	Parameters
Jaya	none (parameter-free)
WOA + best-so-far	$a : 2 \to 0, b = 1$
GWO + best-so-far	$a : 2 \to 0$
PSO	$w = 0.7, c_1 = c_2 = 1.5$
JWOA (Stage 3 off)	$D_{elite} = 0.3, D_{restart} = 0, \beta = 1.5$
LJWOA (Stage 3 on)	$D_{elite} = 0.3, D_{restart} = 0.1, \beta = 1.5$

Table 3. Algorithm configurations

5.1. Additional Experimental Protocol

Instances are drawn from the distributions given above with a different random generator, so absolute values differ from Tables 6–7 and must not be compared across the two groups of tables. As a consistency check we repeated the original protocol (150 iterations for each algorithm, 30 runs): the re-implementation reproduces the significant superiority of LJWOA over Jaya and GWO at every scale and the significant loss to WOA at 200 and 300 microservices, but it does not reproduce the advantage over WOA at 50 and 100 microservices (uncorrected p = 0.080 and 0.262) or the advantage over PSO (significant only at 200 microservices).

Equal evaluation budget. LJWOA with $D_{elite} = 0.3$, $D_{restart} = 0.1$, P = 30 and $I_{max} = 150$ uses $30 + 150(30 + 9 + 3) = 6330$ function evaluations (FE). In the equal-budget experiments every algorithm receives exactly 6330 FE (benchmarks: 8430 FE with $I_{max} = 200$); WOA therefore runs 210 iterations, i.e. 1.4× its original iteration count. Additional baselines: HHO, SMA, L-SHADE (linear population reduction, initial population 40, memory size 6), a real-coded GA (tournament size 3, BLX-0.5 crossover, Gaussian mutation, 2 elites), and NSGA-II (four objectives $\hat{T}, \hat{P}, \hat{G}, B$; the reported value is the best weighted F among all evaluated solutions). All other algorithms use population 30. HHO and SMA schedule their control parameters by the fraction of the budget used.

Held-out tuning of $D_{restart}$. $D_{restart} \in \{0, 0.05, 0.1, 0.15, 0.2, 0.3\}$ is evaluated on 15 tuning instances (disjoint random seeds); the best value is then reported on the 30 evaluation instances.

Statistics. 30 runs per setting; two-sided Wilcoxon signed-rank tests paired by run seed, with Holm–Bonferroni correction [29] within each family of comparisons.

Reference solutions. For 50 microservices (3 instances) a MILP with binary assignment variables, linearised makespan, a big-M encoding of the power cap $\min(u_k, 1)$, and the load-balance term B replaced by the mean absolute deviation of u_k (a lower bound on the standard deviation) was solved with HiGHS under a 100 s time limit. The true F of the incumbent is reported together with the solver's dual bound.

6. Results and Discussion

6.1. Benchmark validation

Table 4 reports mean ± std of best fitness over 20 runs; "JWOA" denotes $D_{restart} = 0$, exactly the two-stage precursor of Section 4.2. Table 5 reports the corresponding Wilcoxon p-values. All statements in this subsection concern JWOA, not the proposed LJWOA configuration; the latter is evaluated on the same functions in Section 6.5. JWOA significantly outperforms Jaya, GWO, and PSO on all six functions. Against WOA (150-iteration budget for WOA, original code) it is nominally better on Sphere (p = 0.0136) and Ackley (p = 0.0046), and statistically tied on the remaining four; after Holm-Bonferroni correction across the 24-test family, only the Ackley result survives – the Sphere result no longer reaches significance. We report the corrected result as the primary claim.

Enabling Stage 3 ($D_{restart} = 0.1$) on the same six functions nominally improves Sphere (does not survive Holm-Bonferroni correction) and significantly degrades Rosenbrock and Schwefel, with no significant effect on Rastrigin, Ackley, or Griewank, indicating the effect reflects a genuine landscape sensitivity rather than a scale-calibration issue. This is why Stage 3 is validated disabled here and enabled separately on the placement problem.

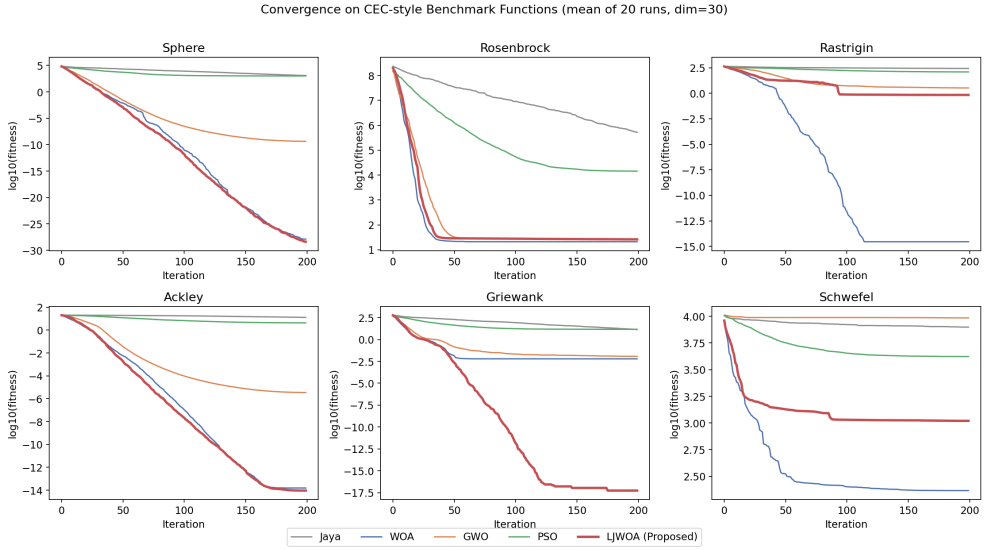


Figure 1. Convergence on the six benchmark functions (mean of 20 runs, dimension 30, log-scale fitness).

6.2. Microservice Placement Results

The results in this subsection are from the original experiments (original implementation, 15 runs). Table 6 reports mean ± std of final weighted fitness $F(x)$ over 15 runs with $D_{restart} = 0.1$; Table 7 reports the corresponding significance tests. LJWOA significantly outperforms Jaya, GWO, and PSO at every scale. Against WOA, LJ-

EARLY VIEW

Function	Jaya	WOA	GWO	PSO	JWOA (Stage 3 off)
F1 Sphere	1.233e+03 ± 352.8	1.275e−28 ± 5.554e−28	4.252e−10 ± 5.457e−10	1.004e+03 ± 2.999e+03	4.118e−29 ± 9.721e−29
F2 Rosenbrock	5.191e+05 ± 2.512e+05	21.05 ± 11.14	27.42 ± 0.634	1.441e+04 ± 3.185e+04	26.15 ± 0.600
F3 Rastrigin	269.7 ± 22.08	2.842e−15 ± 1.239e−14	3.303 ± 2.194	122.8 ± 27.24	0.686 ± 2.989
F4 Ackley	12.96 ± 4.197	1.537e−14 ± 7.587e−15	3.483e−06 ± 1.478e−06	4.292 ± 2.795	9.148e−15 ± 3.806e−15
F5 Griewank	13.61 ± 4.620	0.00612 ± 0.02666	0.01179 ± 0.01473	14.35 ± 32.13	5.551e−18 ± 2.420e−17
F6 Schwefel	7.921e+03 ± 723.4	232.4 ± 430.7	9.665e+03 ± 311.0	4.196e+03 ± 717.7	1.047e+03 ± 1.594e+03

Table 4. Benchmark results – mean ± std of best fitness (20 runs)

[ht]

Function	vs. Jaya	vs. WOA	vs. GWO	vs. PSO
F1 Sphere	1.91e-06 (Y)	1.36e-02 (Y)	1.91e-06 (Y)	1.91e-06 (Y)
F2 Rosenbrock	1.91e-06 (Y)	6.22e-01 (N)	2.67e-05 (Y)	1.91e-06 (Y)
F3 Rastrigin	1.91e-06 (Y)	1.80e-01 (N)	7.08e-04 (Y)	1.91e-06 (Y)
F4 Ackley	1.91e-06 (Y)	4.58e-03 (Y)	1.91e-06 (Y)	1.91e-06 (Y)
F5 Griewank	1.91e-06 (Y)	3.17e-01 (N)	1.91e-06 (Y)	1.91e-06 (Y)
F6 Schwefel	1.91e-06 (Y)	4.75e-01 (N)	1.91e-06 (Y)	2.67e-05 (Y)

Table 5. Wilcoxon p-values, LJWOA (Stage 3 off) vs. baselines on benchmarks (20 runs; Y = significant at $p < 0.05$)

[ht]

Microservices	Jaya	WOA	GWO	PSO	LJWOA (Proposed)
50	0.4074 ± 0.0278	0.3740 ± 0.0300	0.3811 ± 0.0276	0.3849 ± 0.0296	0.3527 ± 0.0245
100	0.4869 ± 0.0221	0.4159 ± 0.0222	0.4582 ± 0.0304	0.4331 ± 0.0312	0.4026 ± 0.0220
150	0.5280 ± 0.0229	0.4402 ± 0.0266	0.4953 ± 0.0305	0.4718 ± 0.0168	0.4376 ± 0.0325
200	0.5434 ± 0.0159	0.4463 ± 0.0325	0.5303 ± 0.0252	0.4923 ± 0.0153	0.4516 ± 0.0177
300	0.6083 ± 0.0164	0.5041 ± 0.0228	0.6215 ± 0.0173	0.5539 ± 0.0178	0.5200 ± 0.0293

Table 6. Placement fitness – mean ± std (15 runs), $D_{restart} = 0.1$

[ht]

Microservices	vs. Jaya	vs. WOA	vs. GWO	vs. PSO
50	6.10e-05 (Y)	1.47e-03 (Y)	1.83e-04 (Y)	6.10e-05 (Y)
100	6.10e-05 (Y)	1.59e-02 (Y)	6.10e-05 (Y)	3.05e-04 (Y)
150	6.10e-05 (Y)	3.30e-01 (N)	1.22e-04 (Y)	2.01e-03 (Y)
200	6.10e-05 (Y)	4.89e-01 (N)	6.10e-05 (Y)	6.10e-05 (Y)
300	6.10e-05 (Y)	8.36e-03 (Y, WOA)	6.10e-05 (Y)	1.16e-03 (Y)

Table 7. Wilcoxon p-values, LJWOA vs. baselines on placement (15 runs; Y = significant, winner noted otherwise)

EARLY VIEW

WOA is significantly better at 50 and 100 microservices, statistically indistinguishable at 150 and 200 (numerically ahead at 150, behind at 200), and significantly behind only at 300 microservices. Unlike the benchmark suite, every one of the 20 placement comparisons retains its significance classification after Holm-Bonferroni correction. These comparisons give WOA 150 iterations, i.e. about 1/1.4 of LJWOA's evaluations; Section 6.6 repeats them at an equal budget.

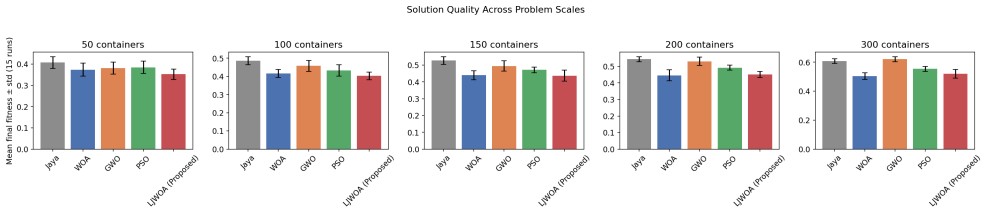


Figure 2. Mean final fitness ± std (15 runs) across problem scales.

6.3. Convergence Speed, Computational Cost, and Ablation

Unlike the two-stage precursor, LJWOA converges more slowly than WOA: measured by iterations to reach 95% of each run's own progress toward its final fitness, WOA reaches this self-normalized threshold in far fewer iterations at every scale (e.g. 43 vs. 134 at 150 microservices, 58 vs. 138 at 300), since Stage 3's unconditional restarts keep perturbing the population late into the run even though the curve ultimately flattens lower. Wall-clock time confirms LJWOA costs 1.36–1.43× WOA's time per run, matching the analytical prediction of Section 4.4: choose LJWOA for better final quality when budget is not binding, plain WOA when decisions must be made in as few iterations as possible.

The original restart-fraction ablation (150 and 300 microservices, $D_{restart} \in \{0, 0.05, 0.1, 0.15, 0.2, 0.3\}$, 15 runs) was run on the same instances that were used for reporting; at 300 microservices every tested $D_{restart} > 0$ beat $D_{restart} = 0$ (0.5307, the two-stage precursor), with 0.1 best (0.5200), and the relationship was not monotonic, degrading again at 0.3. Because this selection used the reporting instances, it is superseded by the held-out tuning of Section 6.7, which should be read as the authoritative evidence for the choice of $D_{restart}$. At 300 microservices, LJWOA also achieved the

lowest communication cost of all five methods and power draw only marginally above WOA's lowest value, while its load-balance standard deviation was second-worst – reflecting the objective's weighting, which favours power and communication cost ($w_2+w_3 = 0.55$) over load balance ($w_4 = 0.10$).

6.4. Overall Ranking

Table 8 summarizes each algorithm's average rank across all 11 test instances. Among the five algorithms of the original study, LJWOA attains the best overall average rank (1.45), narrowly ahead of WOA (1.55); the two are tied on the benchmark suite (1.50 each) and LJWOA leads on placement (1.40 vs. 1.60). Both are far ahead of GWO, PSO, and Jaya (ranks 3.6–4.7). This descriptive ranking should not be read as an omnibus significance test, since the 11 instances combine two problem families and two LJWOA configurations, and it does not extend to the stronger baselines added later: in the equal-budget experiments of Section 6.6 LJWOA's average rank over the five placement scales is 4.8 of 10 algorithms, behind GA (1.2), L-SHADE (1.8), and WOA (3.0).

6.5. Proposed Configuration on the Benchmark Suite

Table 9 evaluates LJWOA with $D_{restart} = 0.1$ (and JWOA) against all baselines at an equal budget of 8430 FE, and Table 10 gives the Holm–Bonferroni-corrected outcome of each comparison (54 tests). LJWOA is statistically tied with WOA on Sphere, Rastrigin, Ackley, and Griewank and significantly worse on Rosenbrock and Schwefel; HHO is significantly better than LJWOA on the same two functions, and L-SHADE and GA are better on Schwefel. Several baselines reach zero (to machine precision) on Rastrigin and Griewank, so those functions do not discriminate. The two-stage JWOA advantage over WOA reported in Table 5 (Sphere, Ackley) is not reproduced once WOA receives the same evaluation budget (280 instead of 200 iterations). The benchmark claims of this paper therefore apply to JWOA relative to Jaya, GWO, and PSO only, and LJWOA is not better than WOA on any benchmark function.

Algorithm	Benchmark avg. rank	Placement avg. rank	Overall avg. rank
LJWOA (Proposed)	1.50	1.40	1.45
WOA	1.50	1.60	1.55
GWO	3.33	4.00	3.64
PSO	4.00	3.20	3.64
Jaya	4.67	4.80	4.73

Table 8. Overall average rank across 11 test instances (descriptive; see Section 5)

6.6. Equal-Budget Placement comparison and Additional Baselines

Table 11 and Fig. 3 report 30-run means at 6330 FE. After Holm–Bonferroni correction (45 tests) LJWOA is significantly better than Jaya, GWO, SMA, and NSGA-II at every scale and tied with PSO at every scale; it is tied with WOA and HHO at 50–200 microservices and significantly worse than both at 300; it is significantly worse than L-SHADE and GA at every scale. Under the original 150-iteration protocol the re-implementation gives a tie with WOA at 50–150 and a significant WOA advantage at 200 and 300 (LJWOA vs. WOA: uncorrected p = 0.080, 0.262, 0.171, 0.0024, $< 10^{-4}$ for 50–300), so the advantage of LJWOA over WOA at 50 and 100 microservices in Table 7 is not reproduced and we no longer claim superiority over WOA, only comparability at 50–200 and a significant loss at 300.

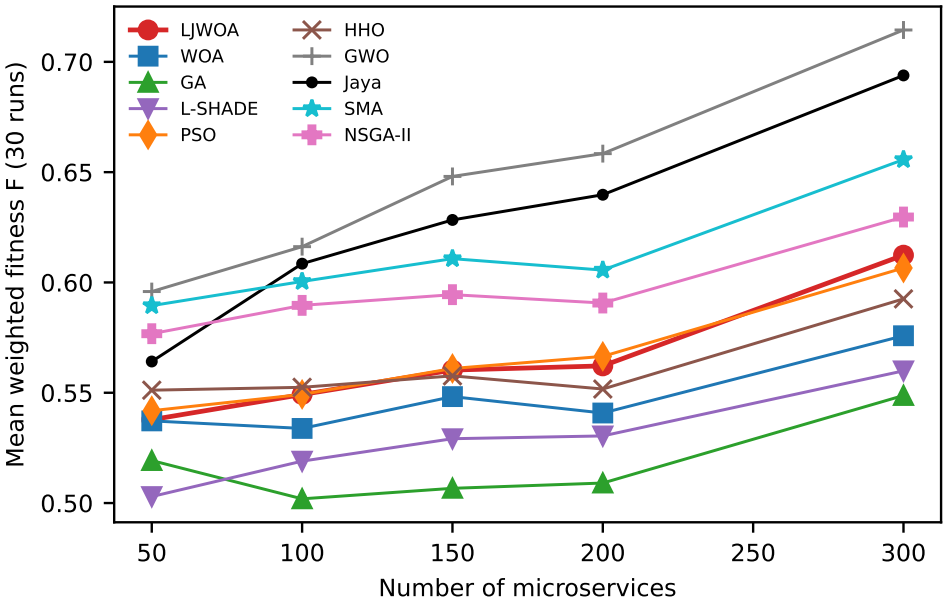


Figure 3. Mean weighted fitness versus number of microservices at equal budget (30 runs, re-implementation).

6.7. Held-Out Tuning of $D_{restart}$

Table 12 shows that the best $D_{restart}$ selected on separate tuning instances varies between 0.05 and 0.3, and that the objective is flat for $D_{restart} \geq 0.05$. On the held-out instances every selected $D^* > 0$ is better than $D_{restart} = 0$ (raw Wilcoxon p < 0.01 at every scale), so Stage 3 improves on the two-stage JWOA. The fixed value 0.1 is within 0.006 of the selected value at 50–200 microservices and 0.013 worse at 300, where the selected $D^* = 0.2$ (0.599) is still significantly worse than WOA (0.576, p = 0.0015). Selecting $D_{restart}$ on the reporting instances, as in the original ablation, is therefore not needed to support the conclusion that Stage 3 helps relative to JWOA, but it does not change the comparison with WOA.

Table 9. Benchmark results at equal budget (8430 FE): mean of best fitness over 30 runs, dimension 30 (re-implementation). LJWOA uses $D_{restart} = 0.1$, JWOA $D_{restart} = 0$.

Function	WOA	GWO	HHO	SMA	L-SHADE	GA	JWOA	LJWOA
F1 Sphere	3.3e-44	1.2e-15	2.8e-38	0	0.018	5.622	1.4e-29	1.8e-29
F2 Rosenbrock	12.6	27.2	0.058	29.0	71.7	633.1	26.5	28.6
F3 Rastrigin	0	0.669	0	0	10.0	29.3	9.464	0
F4 Ackley	3.2e-15	5.8e-9	4.4e-16	4.4e-16	0.913	1.282	1.1e-14	7.4e-15
F5 Griewank	0	6.8e-3	0	0	0.048	1.049	0	0
F6 Schwefel	145.4	9649.3	140.3	4908.5	309.4	1872.6	1432.0	2911.8

Table 10. Outcome of LJWOA ($D_{restart} = 0.1$) versus each baseline on the benchmarks at equal budget (30 runs, Holm–Bonferroni over 54 tests): W = LJWOA significantly better, L = significantly worse, – = no significant difference.

Function	Jaya	WOA	GWO	PSO	HHO	SMA	L-SHADE	GA	JWOA
F1 Sphere	W	–	–	W	–	–	W	W	–
F2 Rosenbrock	W	L	L	W	L	W	W	W	L
F3 Rastrigin	W	–	W	W	–	–	W	W	–
F4 Ackley	W	–	W	W	–	–	W	W	–
F5 Griewank	W	–	W	W	–	–	W	W	–
F6 Schwefel	W	L	W	W	L	W	L	L	L

Table 11. Placement fitness at equal budget (6330 FE): mean ± std over 30 runs (re-implementation); WOA runs 210 iterations. Best per row in bold.

m	Jaya	WOA	GWO	PSO	HHO	SMA	L-SHADE	GA	NSGA-II	LJWOA
50	0.564 ± 0.031	0.537 ± 0.024	0.596 ± 0.050	0.542 ± 0.033	0.551 ± 0.028	0.589 ± 0.045	**0.503** ± 0.027	0.519 ± 0.031	0.577 ± 0.032	0.538 ± 0.029
100	0.609 ± 0.030	0.534 ± 0.027	0.616 ± 0.045	0.549 ± 0.028	0.552 ± 0.031	0.600 ± 0.031	0.519 ± 0.018	**0.502** ± 0.021	0.590 ± 0.024	0.549 ± 0.038
150	0.628 ± 0.029	0.548 ± 0.024	0.648 ± 0.030	0.561 ± 0.025	0.558 ± 0.022	0.611 ± 0.025	0.529 ± 0.012	**0.507** ± 0.017	0.594 ± 0.016	0.560 ± 0.024
200	0.640 ± 0.033	0.541 ± 0.032	0.658 ± 0.040	0.566 ± 0.025	0.552 ± 0.030	0.606 ± 0.032	0.530 ± 0.014	**0.509** ± 0.018	0.591 ± 0.024	0.562 ± 0.036
300	0.694 ± 0.032	0.576 ± 0.028	0.714 ± 0.034	0.607 ± 0.023	0.593 ± 0.029	0.656 ± 0.032	0.560 ± 0.015	**0.549** ± 0.019	0.630 ± 0.021	0.612 ± 0.027

Table 12. Selection of $D_{restart}$ on 15 tuning instances and evaluation on 30 held-out instances (mean fitness). p: raw Wilcoxon test of D^* versus $D_{restart} = 0$ on the held-out instances.

m	selected D^*	D^* held-out	$D = 0.1$ held-out	$D = 0$ held-out	p (D^* vs. 0)	WOA	p (D^* vs. WOA)
50	0.1	0.538	0.538	0.558	0.0000	0.537	0.6850
100	0.3	0.548	0.549	0.573	0.0000	0.534	0.0262
150	0.05	0.566	0.560	0.585	0.0071	0.548	0.0007
200	0.2	0.560	0.562	0.590	0.0000	0.541	0.0345
300	0.2	0.599	0.612	0.624	0.0015	0.576	0.0015

EARLY VIEW

6.8. Stage-Order Ablation

All six orderings of the three stages (W = WOA, J = Jaya, L = Lévy restart) were run at an equal budget (30 runs per scale; Table 13). Differences between orderings are small and not systematic: at 50–200 microservices no ordering differs significantly from the proposed W–J–L (smallest raw p = 0.28), and at 300 microservices W–J–L is the worst ordering (0.612 vs. 0.602 for L–W–J, p = 0.004). We therefore do not claim that the stage order is important; the contribution lies in the combination of the three operators rather than in their sequence, which also means the novelty relative to Jaya–Lévy [16] and WOA–Lévy [17] hybrids is incremental.

m	WJL	JWL	LWJ	WLJ	JLW	LJW	p
50	0.538	0.534	0.536	0.536	0.538	0.539	0.318 (JWL)
100	0.549	0.552	0.550	0.553	0.551	0.550	0.952 (LWJ)
150	0.560	0.558	0.564	0.558	0.562	0.558	0.641 (JWL)
200	0.562	0.568	0.567	0.558	0.563	0.558	0.280 (LJW)
300	0.612	0.607	0.602	0.609	0.605	0.605	0.004 (LWJ)

Table 13. Stage-order ablation (mean fitness, 30 runs, equal budget). W = WOA, J = Jaya, L = Lévy restart; WJL is the proposed order. p: raw Wilcoxon test of WJL against the best other order.

6.9. Capacity Violations

Because capacity is enforced softly, Table 14 reports for the best solution of each run the fraction of overloaded nodes, the total excess utilisation $\sum_k \max(u_k - 1, 0)$, and the maximum utilisation. All algorithms overload most nodes (on average 53–91% depending on algorithm and scale) and the generated instances are oversubscribed in aggregate (at 300 microservices total demand is roughly twice total capacity). Violations grow with scale. At 300 microservices LJWOA overloads more nodes than WOA (84% vs. 65%) with a smaller maximum utilisation (5.5 vs. 5.9). Results should therefore be read as solutions of the soft-constrained model rather than as feasible deployments; a hard-constrained or repair-based variant is future work.

6.10. Weight Sensitivity and Pareto View

The objective weights were varied over five additional vectors (equal; makespan-, power-, latency-, and balance-heavy) at 50, 150, and 300 microservices (30 runs each; Table 15). The comparison between LJWOA and WOA depends on the weights: at 50 microservices LJWOA is significantly better than WOA for all five alternative vectors (Holm, 18 tests); at 150 it is better for two; at 300 it is significantly worse for the base and power-heavy vectors and tied

Table 14. Capacity violations of the final solutions (mean over 30 runs): fraction of overloaded nodes / total excess utilisation $\sum_k \max(u_k - 1, 0)$ / maximum utilisation $\max_k u_k$.

m	WOA	GWO	L-SHADE	GA	LJWOA
50	0.53 / 5.1 / 2.15	0.66 / 14.2 / 3.33	0.73 / 5.2 / 1.73	0.75 / 7.6 / 2.00	0.67 / 5.9 / 1.98
100	0.63 / 17.8 / 3.53	0.68 / 32.0 / 5.33	0.82 / 16.8 / 2.47	0.78 / 21.5 / 2.80	0.75 / 19.5 / 3.04
150	0.62 / 28.7 / 4.40	0.70 / 45.8 / 6.14	0.84 / 29.0 / 2.98	0.79 / 36.4 / 3.30	0.78 / 30.9 / 3.58
200	0.62 / 36.4 / 4.73	0.74 / 53.7 / 6.10	0.84 / 39.7 / 3.24	0.80 / 48.3 / 3.49	0.76 / 50.7 / 4.46
300	0.65 / 56.5 / 5.86	0.80 / 89.5 / 8.16	0.91 / 63.9 / 4.03	0.84 / 80.1 / 4.46	0.84 / 77.4 / 5.54

Table 15. Weight sensitivity: mean fitness difference LJWOA − WOA (negative favours LJWOA, 30 runs); * significant after Holm–Bonferroni correction (18 tests). Last column: rank of LJWOA among the seven algorithms (WOA, Jaya, GWO, PSO, GA, L-SHADE, LJWOA) per weight vector.

m	base	equal	makespan	power	latency	balance	LJWOA rank (6 vectors)
50	+0.001	-0.053*	-0.031*	-0.012*	-0.018*	-0.073*	4,3,2,1,4,3
150	+0.012	-0.053*	-0.012	+0.007	-0.009	-0.079*	4,3,3,2,4,3
300	+0.037*	+0.001	+0.012	+0.067*	+0.003	+0.018	5,4,4,3,5,4

otherwise. L-SHADE and GA obtain the lowest weighted objective for most vectors, and LJWOA ranks between 1st and 5th of seven. Fig. 6 shows the final solutions of 30 runs in the $(\hat{T}, \hat{P})$ and $(\hat{G}, \hat{P})$ planes with the NSGA-II fronts at 150 microservices.

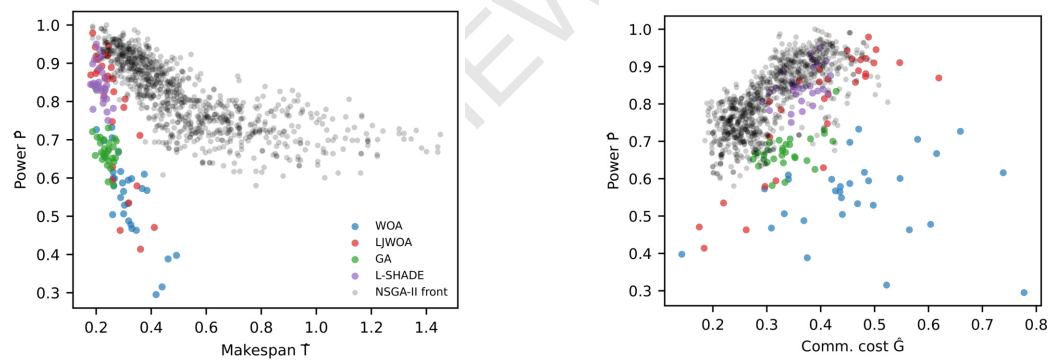


Figure 4. Final solutions of 30 runs (150 microservices) in objective space, with NSGA-II fronts.

6.11. Comparison with an Exact Reference

Table 16 compares the heuristics with the MILP incumbent at 50 microservices (3 instances). The solver did not prove optimality within 100 s (gap 27–29%), so the incumbent is a reference solution, not a proven optimum; its true objective is nevertheless lower than the mean of every heuristic on all three instances (on average LJWOA is 11% above the incumbent, L-SHADE 5%, GA 7%, and WOA 14%).

Instance	MILP	bound	gap	WOA	LJWOA	GA	L-SHADE
1	0.450	0.317	27%	0.513	0.501	0.486	0.483
2	0.468	0.333	28%	0.534	0.520	0.497	0.490
3	0.482	0.341	29%	0.545	0.532	0.521	0.504

Table 16. MILP incumbent (true F), solver dual bound and gap versus the mean of 10 heuristic runs (equal budget), 50 microservices, 100 s limit.

6.12. Convergence Variance and population Diversity

Fig. 4 shows mean ± std best-so-far curves over 30 runs at 150 and 300 microservices. Table 17 confirms that LJWOA needs about twice as large a share of its budget as WOA to reach 95% of a run's own progress.

To examine the loss at 300 microservices, Fig. 5 plots the mean distance of the population from its centroid (normalised by the search-space diagonal). Contrary to the expectation that the restart stage preserves diversity, LJWOA's diversity collapses earlier than WOA's (iteration 50: 0.021 vs. 0.151 at 300 microservices; 0.018 vs. 0.144 at 150): Stages 1–2 use greedy acceptance and Jaya refinement contracts the elite towards the best solution, while the restart replaces only 10% of the population. Premature contraction in the 300-dimensional assignment space is consistent with the loss to WOA, but we have not isolated it as the sole cause, and larger restart fractions did not close the gap (Table 12).

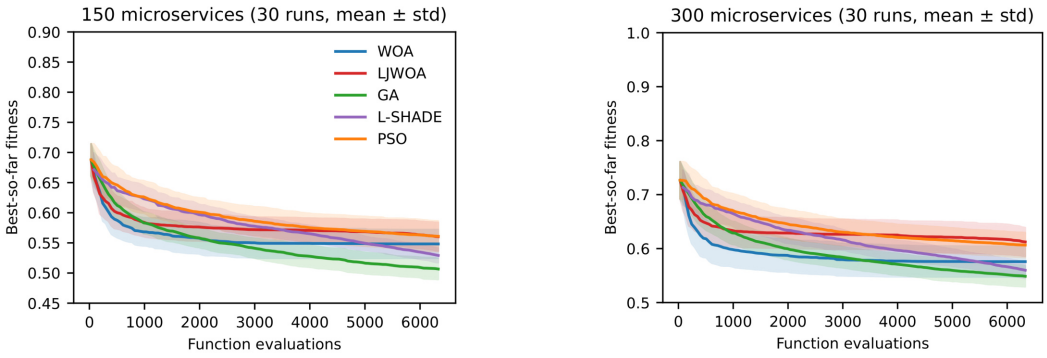


Figure 5. Convergence (mean ± std, 30 runs, equal budget).

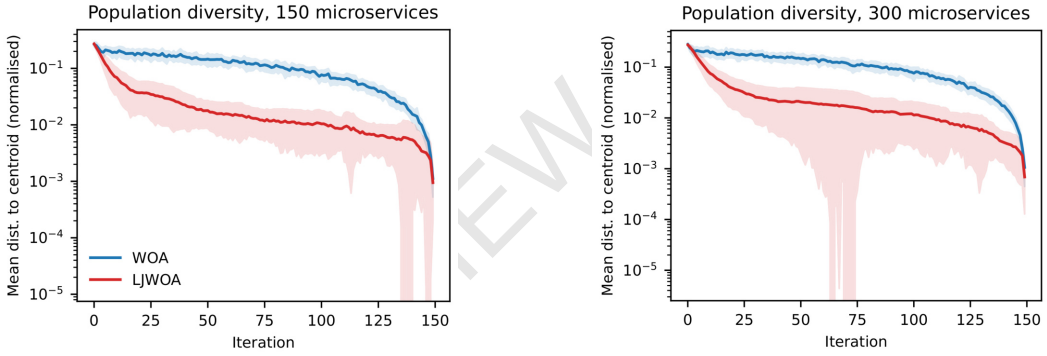


Figure 6. Population diversity of WOA and LJWOA (30 runs, 150 iterations).

Table 17. Convergence speed and population diversity (30 runs, original 150-iteration protocol for diversity, equal budget for convergence). τ_{95}: percentage of the evaluation budget needed to reach 95% of a run's own progress; diversity: mean normalised distance to the population centroid.

m	τ_{95} (%)		Diversity, iter. 50		Diversity, iter. 150	
	WOA	LJWOA	WOA	LJWOA	WOA	LJWOA
50	19	53	0.148	0.020	0.0016	0.0020
100	30	70	0.147	0.026	0.0013	0.0011
150	28	61	0.144	0.018	0.0011	0.0009
200	23	77	0.146	0.018	0.0012	0.0008
300	34	66	0.151	0.021	0.0011	0.0007

7. Conclusion and Future Work

This paper proposed LJWOA, a three-stage hybrid metaheuristic (WOA exploration, Jaya-style elitist refinement, and a Lévy-flight restart of the weakest individuals) for latency- and power-aware microservice placement in fog-cloud environments, and evaluated it on six benchmark functions and synthetic placement instances with 50–300 microservices. The main outcomes are as follows. (i) LJWOA consistently improves on Jaya and GWO and, at an equal evaluation budget, on SMA and NSGA-II; its advantage over PSO reported in the original experiments is not reproduced in the re-implementation. (ii) The Lévy stage improves on the two-stage variant on held-out instances at every scale. (iii) LJWOA is comparable to WOA at 50–200 microservices and significantly worse at 300; the advantage at 50 and 100 microservices observed in the original experiments does not survive an equal evaluation budget in the re-implementation. (iv) GA and L-SHADE obtain lower objective values than LJWOA at every scale, and at 50 microservices the MILP incumbent is on average about 11% better than LJWOA's mean. (v) The stage order is not critical, and the cost is 1.36–1.43× WOA's time per iteration with slower self-normalised convergence. The significance of the work is thus a documented and bounded design study: the Lévy restart is a useful but modest addition to WOA–Jaya hybrids, and it should not be expected to outperform stronger evolutionary baselines on this problem.

This study has several limitations: synthetic instances rather than production traces; a softly enforced node-capacity constraint under which 53–91% of nodes are overloaded; a weighted-sum objective; use of a re-implementation for the additional experiments, whose instance generator differs from the original; a MILP with a linearised balance term and a time limit that did not prove optimality; the 300-microservice loss to WOA, unresolved by the interventions we tried (adaptive elite-fraction scheduling, full-population Jaya refinement, larger populations, longer budgets, larger restart fractions); and no modelling of memory, storage, bandwidth, startup cost, or inter-microservice dependencies. Future work includes validation on real Kubernetes traces, a hard-constrained repair-operator variant, a

EARLY VIEW

traffic-graph-aware communication term, a full Pareto-based multi-objective comparison, a diversity-preserving variant targeting the large-scale loss, and deployment on a physical or emulated testbed.

REFERENCES

[1] Keshri R, Vidyarthi DP. Communication-aware, energy-efficient VM placement in cloud data center using ant colony optimization. International Journal of Information Technology 2023;15(8):4529–4535.

[2] Keshri R, Vidyarthi DP. Energy-efficient communication-aware VM placement in cloud datacenter using hybrid ACO–GWO. Cluster Computing 2024;27(9):13047–13074.

[3] Keshri R, Vidyarthi DP. An ML-based task clustering and placement using hybrid Jaya-gray wolf optimization in fog-cloud ecosystem. Concurrency and Computation: Practice and Experience 2024;36(14):e8109.

[4] Keshri R, Anand N, Vidyarthi DP. Energy-efficient communication-aware VM placement in cloud datacenter using JAYA. Sustainable Computing: Informatics and Systems 2026:101410.

[5] Keshri R, Singh S, Vidyarthi DP. Fog device deployment for enhanced edge communication and network cohesion using JAYA. Journal of Grid Computing 2026;24(2):12.

[6] Supraja T, Chawla P, Gill SS. AI-driven service placement in fog and edge computing environments: a systematic review, taxonomy and future directions. Cluster Computing 2025;28:1070.

[7] Dogani J, Yazdanpanah A, Zare A, et al. A two-tier multi-objective service placement in container-based fog-cloud computing platforms. Cluster Computing 2024;27:4491–4514.

[8] Deng W, Zhu L, Shen Y, et al. A novel multi-objective optimized DAG task scheduling strategy for fog computing based on container migration mechanism. Wireless Networks 2025;31:1005–1019.

[9] Naouri A, Nouri NA, Khelloufi A, et al. Efficient fog node placement using nature-inspired metaheuristic for IoT applications. Cluster Computing 2024;27:8225-8241.

[10] Ghasemi A. MOHHO: multi-objective Harris hawks optimization algorithm for service placement in fog computing. The Journal of Supercomputing 2024;80(17):25004–25028.

[11] Rao RV. Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. International Journal of Industrial Engineering Computations 2016;7(1):19–34.

[12] Mirjalili S, Lewis A. The whale optimization algorithm. Advances in Engineering Software 2016;95:51–67.

[13] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Advances in Engineering Software 2014;69:46–61.

[14] Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of ICNN'95 – International Conference on Neural Networks, vol. 4; 1995, p. 1942–1948.

[15] Yang X-S, Deb S. Cuckoo search via Lévy flights. In: 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC); 2009, p. 210–214.

[16] Vadde S, Kompalli VS. Energy efficient service placement in fog computing. PeerJ Computer Science 2022;8:e1035.

[17] Ling Y, Zhou Y, Luo Q. Lévy flight trajectory-based whale optimization algorithm for global optimization. IEEE Access 2017;5:6168–6186.

[18] Pallewatta S, Kostakos V, Buyya R. Placement of microservices-based IoT applications in fog computing: a taxonomy and future directions. ACM Computing Surveys 2023;55(14s):321:1–321:43.

[19] Pallewatta S, Kostakos V, Buyya R. MicroFog: a framework for scalable placement of microservices-based IoT applications in federated fog environments. Journal of Systems and Software 2024;209:111910.

[20] Hossain MR, Whaiduzzaman M, Barros A, Fidge C. Dynamic microservice placement in multi-tier fog networks. Internet of Things 2024;26:101224.

[21] Kallel A, Rekik M, Khemakhem M. A deep reinforcement learning-based optimization approach for containerized microservice scheduling in hybrid fog/cloud environments. Engineering Applications of Artificial Intelligence 2025;141:109745.

[22] Khan MA, Sahoo B, Mishra SK, Shankar A. Multi-objective based container placement strategy in CaaS. Software: Practice and Experience 2025;55(3):448–472.

[23] Zhu J, Bai W, Zhang H, Lin W, Zhou T, Li K. Adaptive multi-objective swarm intelligence for containerized microservice deployment. Future Generation Computer Systems 2026;174:108012.

[24] Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H. Harris hawks optimization: algorithm and applications. Future Generation Computer Systems 2019;97:849–872.

[25] Li S, Chen H, Wang M, Heidari AA, Mirjalili S. Slime mould algorithm: a new method for stochastic optimization. Future Generation Computer Systems 2020;111:300–323.

[26] Tanabe R, Fukunaga AS. Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation (CEC); 2014, p. 1658–1665.

[27] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 2002;6(2):182–197.

[28] Mantegna RN. Fast, accurate algorithm for numerical simulation of Lévy stable stochastic processes. Physical Review E 1994;49(5):4677–4683.

[29] Holm S. A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics 1979;6(2):65–70.

EARLY VIEW

Author contacts

	Contact
1. Nikhil Anand*	
2. Rohan Kukreti	
3. Prince Kumar Raj	

* Correspondence

Contribution matrix

CRediT role	1	2	3
Conceptualization	•		
Methodology			•
Software			•
Validation	•	•	
Formal analysis	•	•	
Investigation			•
Resources			•
Data curation	•	•	
Writing - original draft	•		
Writing - review and editing	•		
Visualization	•	•	

• Supplied role; blank: assignment absent from this record. Column numbers follow article byline order.

EARLY VIEW

Declarations

Generative AI

The authors did not use generative artificial intelligence or AI-assisted tools to generate the text, analysis, results, figures, or conclusions of this manuscript.

Funding

The authors received no specific funding for this research from any public, commercial, or not-for-profit funding agency.

Competing interests

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

Data availability

The study uses synthetically generated fog–cloud microservice-placement instances and algorithmic benchmark functions. The data-generation procedures and experimental settings are described in the manuscript. The corresponding data and implementation materials are available from the corresponding author upon reasonable request.

Ethics approval

Not applicable. This study did not involve human participants, human data, or animals.

Consent

Not applicable.

668 **Trial registration**

669 Not applicable.

EARLY VIEW