Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 20 No. 21 (2026) | OPEN ACCESS

https://doi.org/10.3991/ijim.v20i21.63140

PAPER

An Intelligent Resource Optimization Framework for Mobile Applications

Esra'a Alhenawi[1], Omar Alidmat[2], Esraa Abu Elsoud[3](✉)

[1]Al al-Bayt University, Mafraq, Jordan

[2]Zarqa University, Zarqa, Jordan

[3]Applied Science Private University, Amman, Jordan

e_abuelsoud@asu.edu.jo

ABSTRACT

Mobile edge computing (MEC) enables latency-sensitive mobile applications to offload computation to nearby edge servers, but selecting an offloading and resource-allocation policy that jointly minimizes energy, latency and deadline violations is an NP-hard combinatorial problem for which exact solvers do not scale. This paper proposes hybrid beam search–snake optimizer (HBS-SO), an intelligent resource-optimization framework that integrates three complementary search mechanisms: a domain-aware beam search (BS) that fixes the discrete offloading structure, the snake optimizer (SO) that explores the resource-allocation space from a beam-seeded population, and a memetic local-search stage that refines the continuous variables (CPU frequency, edge-server selection and transmit power). We evaluate HBS-SO on an Alibaba-calibrated mobile workload under two problem formulations: an offloading-centric formulation with 2N decision variables (offload decision and CPU frequency) and a multi-resource formulation with 4N decision variables (adding edge-server selection and transmit power). Against BS, the SO and particle swarm optimization (PSO) over 30 independent runs with paired Wilcoxon testing, HBS-SO ranks first on weighted fitness in both formulations. In the offloading-centric formulation it significantly outperforms every baseline, including PSO (+18.0%, $p < 0.001$), while simultaneously attaining the lowest energy consumption (814 J versus roughly 1300 J, a 38% reduction) and the lowest worst-case latency (16.0 s versus 22–29 s, a 27–37% reduction). In the multi-resource formulation, it again ranks first on weighted fitness—significantly ahead of BS (+47.1%) and SO (+14.5%) and statistically on par with the strongest baseline PSO—while also delivering the lowest energy and worst-case latency and the highest deadline-satisfaction rate. We state the PSO comparison plainly: in the multi-resource setting the two methods are statistically indistinguishable on the objective, and HBS-SO's advantage there lies in the operational metrics. HBS-SO is also the most stable of the stochastic optimizers across both settings. Overall, HBS-SO provides a strong and dependable optimization approach for energy- and latency-critical mobile edge deployments.

KEYWORDS

mobile edge computing (MEC), task offloading, snake optimizer (SO), beam search (BS), hybrid metaheuristic, tail latency, resource allocation, predictability

Alhenawi, E., Alidmat, O., Elsoud, E. A. (2026). An Intelligent Resource Optimization Framework for Mobile Applications. *International Journal of Interactive Mobile Technologies (iJIM)*, 20(21), pp. 40–60. https://doi.org/10.3991/ijim.v20i21.63140

Article submitted 2026-08-05. Revision uploaded 2026-09-20. Final acceptance 2026-09-23.

© 2026 by the authors of this article. Published under CC-BY.

1 INTRODUCTION

The proliferation of computationally demanding yet delay-sensitive mobile applications—augmented reality, real-time video analytics, interactive gaming and on-device inference—has outpaced the energy and compute budgets of handheld devices. Mobile edge computing (MEC) addresses this gap by placing compute resources at the network edge so that tasks can be offloaded from the device to nearby servers, reducing both execution latency and device energy consumption [1] [2]. The central control problem is to decide, for each task, whether to execute locally or offload, and simultaneously to allocate the associated resources processor frequency and, in richer formulations, the target edge server and the wireless transmit power [3]. Because these decisions couple a discrete offloading choice with continuous resource variables under deadline constraints, the resulting optimization is NP-hard, and exact solvers do not scale to the hundreds of tasks encountered in practice. Metaheuristic optimizers particle swarm optimization (PSO), genetic algorithms, and a growing family of nature-inspired algorithms are the dominant practical response to this intractability, and they routinely deliver good offloading policies at acceptable cost [4] [5]. More recently, deep reinforcement learning (DRL) has been applied to the same problem with strong results [6]. However, most of this literature evaluates a single best-of-run solution and reports mean performance, while the run-to-run variability of stochastic optimizers is seldom quantified. For a deployed MEC scheduler that is invoked repeatedly, this variability matters as much as average quality: a policy generator whose output quality swings widely between invocations is difficult to provision for, particularly when the operative service-level objective is a bound on worst-case (tail) latency rather than mean latency [7].

This paper proposes hybrid beam search–snake optimizer (HBS-SO), an intelligent resource-optimization framework built around the observation that the offloading problem has two distinct sub-structures a discrete offload pattern and a continuous resource allocation that reward different search strategies. HBS-SO addresses them with three mechanisms. First, beam search (BS)—a classical bounded-width search—fixes a strong discrete offloading skeleton by exploiting the combinatorial structure of the offload decision. Second, the SO, a recent nature-inspired metaheuristic [8], explores the resource allocation space from a population seeded with the beam solutions, inheriting BS's structural competence while retaining global search ability. Third, a memetic local search stage refines the continuous variables—CPU frequency, edge-server selection and transmit power—around the evolved solution, tightening the allocation that a purely population-based optimizer tunes only coarsely. In our experiments,, this last stage turned out to matter more than we initially expected: without it the hybrid was only competitive with a well-tuned PSO, whereas the local refinement is what lets it consistently reach the best solutions. To answer this, we evaluate HBS-SO on the same underlying mobile workload under two complementary formulations—an offloading-centric formulation with 2N decision variables and a multi-resource formulation with 4N decision variables—and compare it against BS, SO, and PSO over 30 independent seeded runs with paired statistical testing. The contributions of this work are:

- An intelligent resource-optimization framework, HBS-SO, that integrates three complementary search mechanisms—BS to fix the discrete offloading structure, the SO to explore the resource-allocation space from a beam-seeded population, and a memetic local-search stage to refine the continuous variables (CPU frequency, edge-server selection, and transmit power) (Sections 4–5).

- A comprehensive empirical study on two problem formulations showing that HBS-SO achieves the best weighted fitness in both settings. In the offloading-centric formulation, it significantly outperforms every baseline, including PSO (+27.3%/+59.8%/+18.0% over BS/SO/PSO, all $p < 0.001$); in the multi-resource formulation it ranks first on weighted fitness, statistically on par with the strongest baseline PSO and significantly ahead of BS and SO (Sections 6–8).
- An analysis of why the hybrid performs well—the continuous-refinement stage resolves the apparent energy–latency trade-off by pairing a low offload ratio with reduced local CPU frequencies—supported by scalability and sensitivity studies that characterize how the advantage behaves with problem size and hyperparameters (Sections 6–9).

The remainder of the paper is organized as follows: Section 2 reviews related work. Section 3 defines the two problem formulations. Section 4 presents the HBS-SO framework. Section 5 describes the experimental setup. Sections 6 and 7 report results for the two formulations, Section 8 provides the cross-setting analysis, and Section 9 reports scalability and sensitivity. Section 10 states limitations and Section 11 concludes.

2 RELATED WORK

2.1 Task offloading formulations in MEC

Computation-offloading problems in MEC span a spectrum of decision richness. Offloading centric formulations decide a binary local/edge assignment and, often, a processor frequency, and are the most studied because they capture the core trade-off between local energy and offload latency [9] [10]. Partial-offloading formulations divide a task between device and edge to exploit finer granularity [11], while multi-resource formulations additionally allocate server selection, bandwidth, or transmit power, producing substantially larger decision spaces [12] [13]. Joint offloading with service caching or content placement further couples the decision variables. Our study deliberately spans this spectrum by evaluating the same algorithms on both a 2N offloading centric formulation and a 4N multi-resource formulation, which lets us isolate how solver behavior changes with formulation richness rather than with the workload.

2.2 Metaheuristic and swarm-based optimizers

Particle swarm optimization and its variants remain the workhorse metaheuristics for offloading. Improved PSO schemes adapt inertia or learning coefficients to the offloading landscape [5], quantum-behaved PSO enlarges the explorable region [11], and integrated PSO couples the swarm with problem specific operators [14]. Beyond PSO, distributed metaheuristic schedulers aim to scale across edge nodes [15], and heuristic offloading with explicit delay constraints trade off some optimality for tight latency control [16]. These methods establish the strong baselines we compare against; we adopt PSO as an external, independently implemented baseline and a three-restart SO as a fair single-metaheuristic reference.

2.3 Hybrid metaheuristics

Hybridization—combining two search mechanisms to offset each other's weaknesses—is an active theme. Recent works hybridize grey-wolf and Jaya operators for joint offloading and resource management [17], fuse swarm intelligence with routing heuristics for energy-aware MEC [18] and combine metaheuristics with SDN-based control for edge resource allocation [19]. Ant colony optimization has been paired with reptile-search operators for offloading [20], and hybrid designs have been applied to energy optimization in 5G MIMO systems [21] and UAV-assisted MEC [22] [23]. Most such studies report a fitness or energy improvement over the constituents. Our contribution differs in two respects: we hybridize a deterministic bounded-width search (Beam Search) with a stochastic metaheuristic rather than combining two stochastic operators, and we add a dedicated continuous-refinement stage that lets the hybrid lead not only on the weighted objective but also on the operational metrics—energy, tail latency and deadline satisfaction—that matter most for deployment.

2.4 Learning-based offloading

Deep reinforcement learning has become a prominent alternative, framing offloading as a sequential decision process. Proximal-policy and actor-critic methods optimize joint energy and latency [24], deadline-aware agents balance offloading against deferral [25], and federated or multi-agent DRL scales the approach across devices while preserving privacy [26] [27]. Lyapunov-guided DRL provides stability guarantees for online offloading [28], and value-based agents target energy efficiency under mobility [29] [30]. DRL methods are powerful but require training, careful reward design, and large interaction budgets; the metaheuristic route we pursue remains attractive when an auditable, training-free policy generator is preferred.

2.5 Energy, latency and reliability objectives

A parallel thread emphasizes the objectives themselves. Latency-aware joint offloading and energy control couples the two goals explicitly [31], and latency-energy optimization has been studied in SCMA-enhanced UAV-MEC networks [32]. Reliability and deadline guarantees are central in vehicular and fog settings, were guaranteed task completion [33], reliable deadline- bounded offloading [34], and delay-sensitive emergency workloads [35] motivate worst-case rather than average metrics. Adaptive AI-enhanced schemes optimize quality of experience [36], and energy-efficient designs recur across UAV-enabled MEC [37] The tail-latency perspective that we adopt is comparatively under-represented in the metaheuristic offloading literature despite its practical importance, and it is the axis on which HBS-SO is most competitive. Recent studies have investigated computation offloading and resource allocation optimization in MEC environments using various artificial intelligence and metaheuristic optimization techniques. Existing approaches have focused on minimizing key performance objectives, including energy consumption, execution latency, and deadline violations. Metaheuristic-based methods, such as Grey Wolf Optimization, JAYA, PSO, and evolutionary algorithms, have demonstrated promising results in addressing the NP-hard nature of MEC optimization problems. However, many existing solutions rely on a single optimization strategy or are evaluated

mainly through simulations, leaving opportunities for developing more adaptive hybrid frameworks that effectively combine exploration and exploitation capabilities. Table 1 summarizes representative studies related to MEC task offloading and resource allocation, highlighting their optimization approaches, objectives, and limitations.

Table 1. Summary of related work on computation offloading and resource allocation in MEC

Reference	Approach	Objective	Key Contribution	Limitation/Gaps
[17]	Hybrid AJEGWO (GWO + JAYA)	Minimize energy and latency	Introduced a hybrid metaheuristic combining GWO and JAYA with adaptive momentum and Levy flight to improve exploration and convergence in MEC offloading.	Validated mainly through simulations; real-world MEC deployment was not investigated.
[38]	Alternating Optimization and Convex Optimization	Reduce end-to-end latency	Proposed joint optimization of offloading ratio, UE/edge CPU frequency, and transmission power for MEC systems.	Limited scalability analysis and evaluation under diverse network conditions.
[39]	Deep Reinforcement Learning (DRL)	Reduce energy consumption and latency	Developed a DRL-based framework for joint offloading and resource allocation in dynamic MEC environments.	Partial offloading optimization under highly dynamic wireless conditions remains limited.
[40]	Adaptive Particle Swarm Optimization	Minimize system energy consumption	Presented cache-assisted MEC optimization considering offloading, CPU allocation, and bandwidth resources.	Designed for a specific caching scenario; generalization to other MEC architectures requires further study.
[41]	Hybrid Energy Efficient Task Offloading Algorithm (HEETA)	Improve energy efficiency and task execution performance	Proposed a hybrid offloading algorithm with improved convergence and robustness under different MEC parameters.	Requires validation across broader MEC environments and workloads.
[42]	Lagrange Duality Resource Optimization Algorithm (LDROA)	Optimize latency and computation cost	Developed an offloading and resource allocation strategy for real-time IoT applications using MEC.	Evaluation focused on specific IoT scenarios with limited generalization.
[43]	NSGA-II with SA-based Mutation	Optimize latency and energy consumption	Introduced a hybrid multi-objective algorithm for UAV-assisted MEC with task dependency and CPU allocation optimization.	Scalability for large-scale UAV-MEC systems remains unexplored.
[44]	Binary Multiobjective Grey Wolf Optimization	Optimize latency, energy, and resource cost	Formulated MEC task offloading as an NP-hard problem and proposed an evolutionary optimization solution.	Computational complexity may increase for large-scale MEC scenarios.

3 SYSTEM MODEL AND PROBLEM FORMULATIONS

We consider a single mobile device with a set of N independent tasks to be scheduled between local execution and a set of heterogeneous edge servers. Each task is characterized by its required CPU cycles ci, memory footprint mi, uplink data size di, and a soft deadline Di. The device CPU frequency is tunable within the range [fmin, fmax] = [0.6, 2.4] GHz. The local dynamic energy consumption follows the standard CMOS model:

$$Elocal = \kappa f 2ci$$

where $\kappa = 10 - 27$ represents the switched-capacitance coefficient. For offloaded tasks, the uplink transmission time is determined according to the Shannon rate:

$$R = Blog\,2(1 + SNR)$$

where B denotes the channel bandwidth. The corresponding transmission of energy is calculated based on the required transmission time and power consumption. The scheduler aims to minimize a weighted objective function that jointly considers energy consumption, worst-case latency, memory constraint violations, and deadline violations. Two different formulations are evaluated based on this unified MEC system model. As illustrated in Figure 1, we consider a single mobile device with a set of N independent tasks to be scheduled between local execution and heterogeneous edge servers.

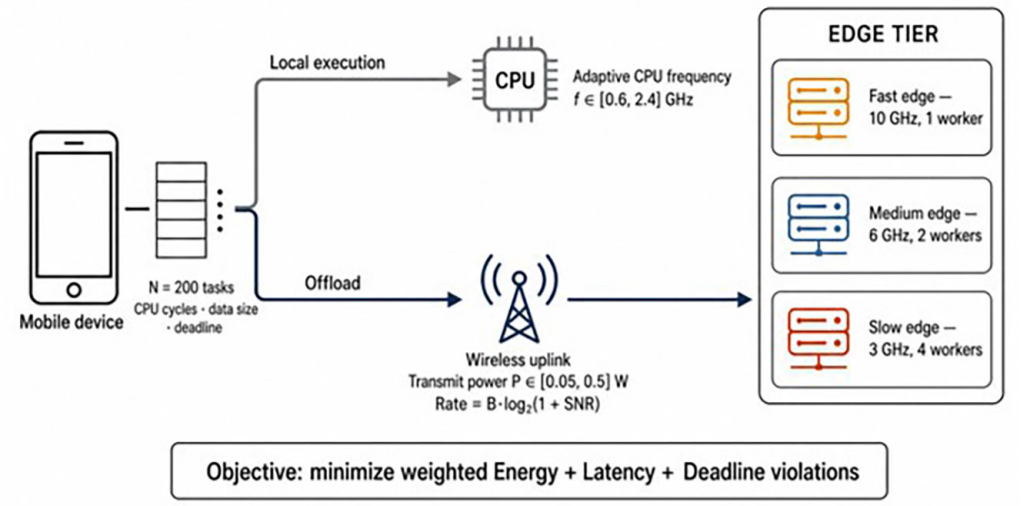


Fig. 1. MEC system model with 200 independent tasks, supporting local execution and wireless offloading to three heterogeneous edge servers

Note: The scheduler jointly optimizes energy, latency, memory, and deadline constraints.

3.1 Setting A—offloading-centric formulation (2N-dimensional)

In the offloading-centric formulation, the decision vector encodes, for each task, a binary offload decision and a normalized CPU frequency, giving 2N dimensions. Offloaded tasks are served by an aggregate edge tier subject to a congestion penalty when the number of concurrent offloads exceeds capacity. This formulation represents the traditional edge-offloading setting in which the discrete offload pattern dominates solution quality, and it is where a structural heuristic such as BS is expected to be strong. The fitness function is defined as:

$$F = wEE^{-} + wLT^{-}max + wMM^{-} + wpp$$

where E^{-}, T^{-}max, and M^{-} represent the normalized energy consumption, worst-case latency, and memory usage terms, respectively, while P denotes the deadline-violation penalty. The weighting coefficients are set as $(wE, wL, wM, wP) = (0.35, 0.35, 0.10, 4.0)$.

3.2 Setting B—multi-resource formulation (4N-dimensional)

The multi-resource formulation augments each task's decision with an edge-server selection (among three heterogeneous servers with distinct clock rates and worker counts) and a continuous transmit power in [0.05, 0.50] W, giving 4N dimensions. Offloaded tasks are queued at the selected server's workers with realistic start-time contention, and the transmission rate depends on the chosen power through the SNR. This formulation represents modern multi-parameter resource management, in which the enlarged continuous search space favors free-form metaheuristic exploration. The fitness function retains the same structure as defined in Equation 3, with the weighting coefficients set as (wE, wL, wM, wD) = (0.35, 0.30, 0.05, 0.30). The last term represents the penalty associated with deadline violations. These weights reflect a deployment stance that treats energy and latency as the primary co-equal objectives, keeps memory a soft secondary concern, and enforces deadlines strongly. Because all four algorithms are compared under identical weights, the relative comparison is unaffected by the specific choice; a full weight-sensitivity study is left as future work. The two optimization settings considered in this work are illustrated in Figure 2. Setting A represents the basic offloading-centric problem, while Setting B extends the decision space by including additional resource allocation variables.

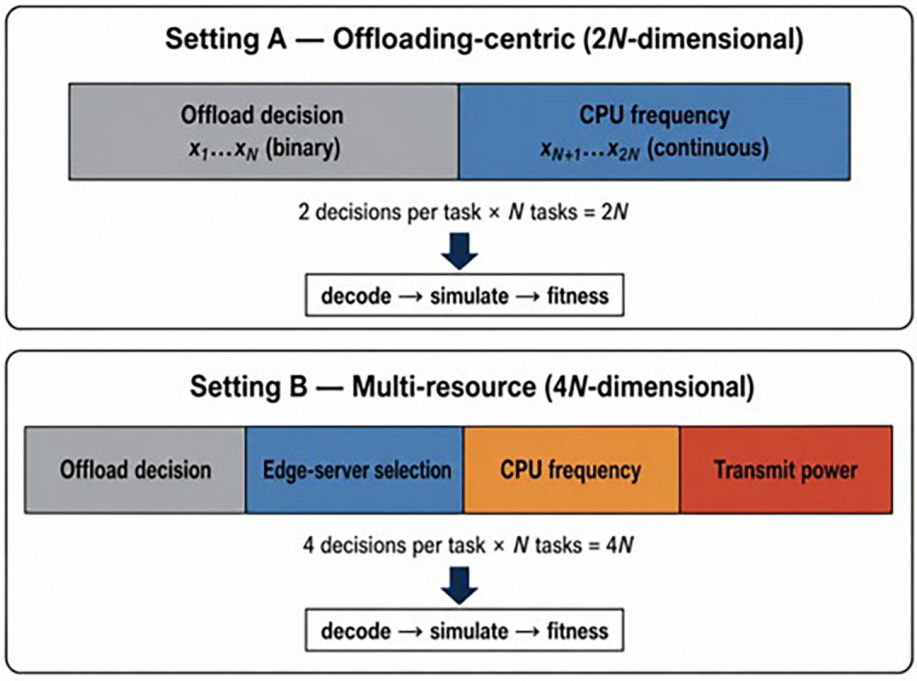


Fig. 2. The optimization settings

4 THE HBS-SO FRAMEWORK

To efficiently solve the formulated MEC optimization problem, we propose a HBS-SO framework that combines discrete decision exploration with continuous resource optimization. The main idea is to first identify promising offloading structures using BS and then refine the associated continuous resource allocation variables using the SO. This hybrid strategy enables effective exploration of the

large mixed discrete-continuous search space while maintaining efficient convergence. The overall workflow of the proposed HBS-SO framework is illustrated in Figure 3.

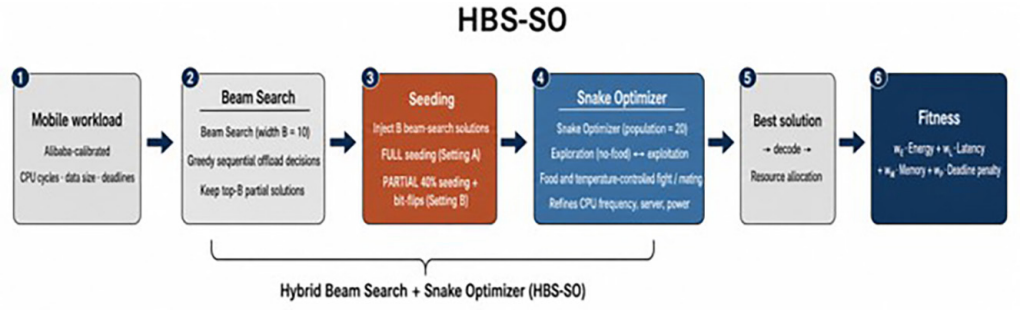


Fig. 3. The HBS-SO framework: Beam Search produces top-B offloading skeletons that seed the SO population (full seeding in Setting A, partial in Setting B); SO refines the continuous variables and the best solution is decoded and scored by the weighted fitness

4.1 Beam search component

Beam search constructs the offload decision sequentially over tasks, maintaining a beam of the B best partial assignments at each step. For each partial solution, it evaluates both possible extensions (local or offload) using a fast surrogate that fixes the continuous variables at their midpoint and scores the resulting configuration under the true fitness, retaining the top B candidates. A small tie-breaking noise avoids degenerate beams. Because the offload pattern is the dominant structural decision, this bounded-width search reaches a near-optimal discrete skeleton at O(N·B) fitness evaluations, far below exhaustive enumeration.

4.2 Snake optimizer component

The SO [8] is a nature-inspired metaheuristic that models snake foraging and mating. The population is split into male and female sub-populations whose search behavior is modulated by two environmental signals, temperature and food quantity, computed from the iteration index. When food is scarce the snakes explore; when food is available, they exploit, and at low temperature they switch between fight and mating modes that recombine the best male and female solutions. We use the standard parameterization (C1, C2, C3) = (0.5, 0.05, 2.0) with temperature and food thresholds (0.6, 0.25). As a fair single-metaheuristic baseline, we also run SO with three random restarts, keeping the best.

4.3 Hybridization: adaptive beam seeding

HBS-SO couples the two components by initializing the SO population with BS solutions rather than random vectors. The degree of seeding is adapted to the formulation. In the offloading-centric formulation, where BS is near-optimal on the discrete skeleton, we use full seeding: the top-B beam solutions populate the initial swarm, with small Gaussian perturbations on the frequency variables so that SO refines the

continuous dimensions around a strong discrete backbone. In the multi-resource formulation, where BS captures only a fraction of the enlarged decision space and can otherwise trap the optimizer in a narrow basin, we use partial seeding: only a fraction (40%) of the population is beam-seeded, with stochastic bit-flips on the offload pattern and perturbations on the continuous variables, while the remainder is random to preserve exploration. Adaptive beam seeding fixes the discrete backbone—exactly where BS is most informative—and leaves the continuous allocation (CPU frequency in Setting A; frequency plus transmit power and edge-server choice in Setting B) to SO exploration.

4.4 Memetic local-search refinement

Beam seeding fixes the discrete backbone, but the SO's population operators tune the continuous variables only coarsely. The final stage therefore refines the best solution returned by SO with an adaptive local search over its continuous dimensions—CPU frequency in Setting A, and CPU frequency together with transmit power in Setting B—while the offload pattern and the edge-server assignment remain fixed. Concretely, the search process is executed for a maximum of 1,500 iterations. At each iteration, the continuous variables are perturbed using Gaussian noise with a step scale σ, initialized as $\sigma 0 = 0.15$. Additionally, with a small probability, a single offloading bit is flipped to explore alternative discrete solutions. A candidate move is accepted only if it strictly decreases the fitness value. The step scale σ is increased after an accepted move and reduced through an annealing mechanism after a rejected move. If the search process becomes stagnant, σ is reset to $\sigma 0$. This adaptive strategy enables the optimizer to alternate between fine-grained exploitation around promising solutions and occasional larger exploratory jumps. Because the perturbations act on the continuous allocation around an already-strong discrete backbone, this refinement concentrates search effort where it is most productive, which is the main structural difference from PSO, whose swarm updates all variables jointly without any such initialization. The refinement operates on a single incumbent solution, so its cost is bounded by the iteration count, and it adds only a modest overhead to the overall run.

5 EXPERIMENTAL SETUP

We evaluate on a synthetic workload of $N = 200$ tasks calibrated to match the statistical properties of the Alibaba Cluster Trace v2018. CPU demand follows a Pareto distribution ($\alpha = 2.5$), memory footprint is log-normally distributed, task data sizes are log-uniform over 100 KB–10 MB, and deadlines are set as the base duration scaled by a uniform slack in [2, 6]. This calibration provides workload realism while ensuring full reproducibility; the same workload instance is used for both formulations so that differences arise from the formulation, not the data. As illustrated in Figure 4. We compare four algorithms—BS, the three-restart SO, PSO, and the proposed HBS-SO. PSO is used as an external baseline through the independent MealPy implementation (OriginalPSO) with its default cognitive and social coefficients, so that it is not tuned with knowledge of our results. The SO uses the parameters listed in Table 2 and is run as a three-restart best-of baseline. Each algorithm is run for 30 independent seeds with a population of 20 and a beam width of 10 (for the beam-search-based methods). The population metaheuristics (SO, PSO) use 60 iterations; the SO stage inside HBS-SO is allotted 120 iterations in Setting B,

because the enlarged 4N continuous space needs more exploration steps for the seeded population to converge. We note this budget difference explicitly: it reflects the higher exploration cost of the multi-resource formulation rather than an advantage granted to HBS-SO alone, and Section 8 confirms that HBS-SO's Setting-B fitness is on par with, not significantly better than PSO. Statistical significance is assessed with the Wilcoxon signed-rank test paired by seed, which makes no normality assumption and controls for seed-induced variance.

The three HBS-SO stages have transparent complexity. BS performs O(N·B) fitness evaluations to build the offloading skeleton; the SO stage costs O (pop · iter · N) as for any population metaheuristic; and the memetic refinement adds a bounded O(iter_ls) evaluations on a single incumbent. HBS-SO therefore carries the cost of its BS initialization and the local-search pass on top of a standard metaheuristic run. In our environment, the full 30-run campaign completed in roughly 130 s for Setting A and 12 min for Setting B on a commodity CPU; per-run wall-clock for HBS-SO is a small constant factor above SO alone, which is acceptable for an offline scheduling decision.

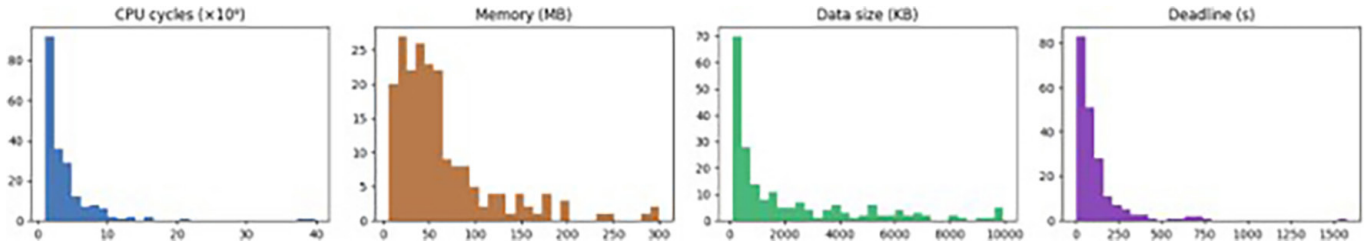


Fig. 4. Distributions of the calibrated workload: CPU cycles, memory, data size, and deadlines (N = 200 tasks)

Table 2. Key simulation and algorithm parameters

Parameter	Setting	Value
Tasks (N)	Both	200
Independent runs	Both	30
Population size	Both	20
SO/PSO iterations	Setting A	60
SO/PSO iterations	Setting B	60
HBS-SO SO iterations	Setting B	120
Beam width (B)	Both	10
Snake Optimizer (C1/C2/C3)	Both	0.5/0.05/2.0
Snake Optimizer thresholds (temperature/food)	Both	0.6/0.25
SO baseline restarts	Both	3 (best-of)
PSO implementation	Both	MealPy OriginalPSO (default c1/c2/w)
Memetic local-search iterations	Both	1500
Memetic step scale (σ0)	Both	0.15
CPU frequency range	Both	0.6–2.4 GHz

(Continued)

Table 2. Key simulation and algorithm parameters *(Continued)*

Parameter	Setting	Value
Transmit power range	Setting B	0.05–0.50 W
Objective weights (E/L/M/P)	Setting A	0.35/0.35/0.10/4.0
Objective weights (E/L/M/D)	Setting B	0.35/0.30/0.05/0.30
Significance test	Both	Paired Wilcoxon signed-rank

6 RESULTS

This section devoted for displaying and discussing the results.

6.1 Setting A (offloading-centric, 2N)

In the offloading-centric formulation, HBS-SO achieves the best performance across the primary evaluation metrics, as presented in Tables 3 and 4. It attains the lowest mean fitness (0.2264 ± 0.0101), significantly outperforming BS (+27.3%), the SO (+59.8%) and PSO (+18.0%), all at $p < 0.001$. The advantage extends well beyond the objective value: HBS-SO simultaneously achieves the lowest energy consumption (814 J, a 38% reduction relative to the 1250–1320 J of the baselines) as shown in Figure 10 and the lowest worst-case latency (16.0 s versus 22.1 s for PSO and 25.2 s for BS, a 27–37% reduction in tail latency), while maintaining a 100% deadline-satisfaction rate as shown in Figure 5. This broad dominance follows directly from the framework's design: the beam skeleton fixes a strong offloading pattern, and the memetic local search then tunes the continuous CPU-frequency allocation to suppress both energy and the latency of the worst-served task as shown in Figure 9.

Table 3. Setting A multi-metric results over 30 runs (mean ± std where applicable)

Algorithm	Fitness (mean ± std)	Fitness (best)	Energy (J)	Lat. Mean (s)	Lat. Max (s)	Deadline OK
BS	0.3114 ± 0.0037	0.3065	1305.3	2.43	25.21	100.0
SO	0.5631 ± 0.1586	0.2719	1251.6	4.18	28.53	98.8
PSO	0.2762 ± 0.0280	0.2355	1321.5	4.49	22.12	99.9
HBS-SO	**0.2264 ± 0.0101**	**0.1998**	**814.5**	3.52	**16.02**	**100.0**

Note: Lower fitness, energy, and latency values indicate better performance.

Table 4. Setting A Wilcoxon signed-rank tests (HBS-SO versus each baseline)

Comparison	W	*p*-Value	Significant	Δ Mean (%)
HBS-SO vs BS	0.00	0.00000	Yes	27.30
HBS-SO vs SO	0.00	0.00000	Yes	59.80
HBS-SO vs PSO	2.00	0.00000	Yes	18.04

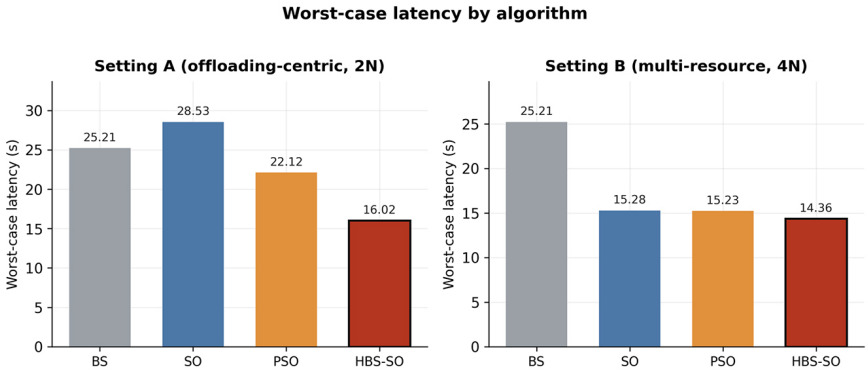


Fig. 5. Worst-case (tail) latency across algorithms in both formulations; lower is better

It is worth explaining why HBS-SO can attain the lowest energy, the lowest worst-case latency, and the lowest offload ratio at the same time, since these three might be expected to trade off. The key is that the objective couple's energy and tail latency through the CPU-frequency and offload variables jointly. HBS-SO offloads a small, carefully chosen 16.2% of tasks, specifically the few compute-heavy jobs whose local execution would dominate the worst-case latency and runs the remaining tasks locally at reduced CPU frequencies. Because dynamic energy scales with the square of frequency, lowering the clock on the many locally executed tasks cuts total energy sharply, while offloading only the latency-critical few is enough to bound the tail. PSO, by contrast, offloads far more aggressively (46.3%), which raises transmission energy without a commensurate latency benefit. The apparent trade-off is thus resolved by a better joint allocation rather than by favoring one metric over another; this is precisely the allocation that the memetic refinement discovers around the beam-fixed backbone. The framework is also the most stable of the stochastic optimizers in this setting. HBS-SO's fitness standard deviation (0.010) is roughly one-third of PSO's (0.028) and less than one-tenth of the Snake Optimizer's (0.159), as shown in Figure 6. The SO alone is competitive at its best (best-of-run 0.272) but highly inconsistent; beam seeding followed by memetic refinement removes this instability, yielding an optimizer whose output quality is tightly bounded across invocations a valuable property for a scheduler invoked repeatedly in deployment. Figure 7 shows the per-run fitness distribution across the 30 runs.

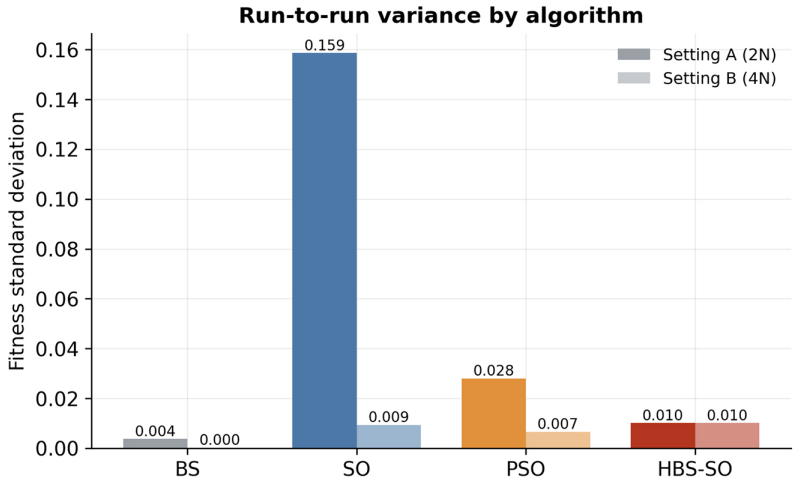


Fig. 6. Run-to-run fitness variance by algorithm. HBS-SO is the most stable stochastic optimizer in Setting A

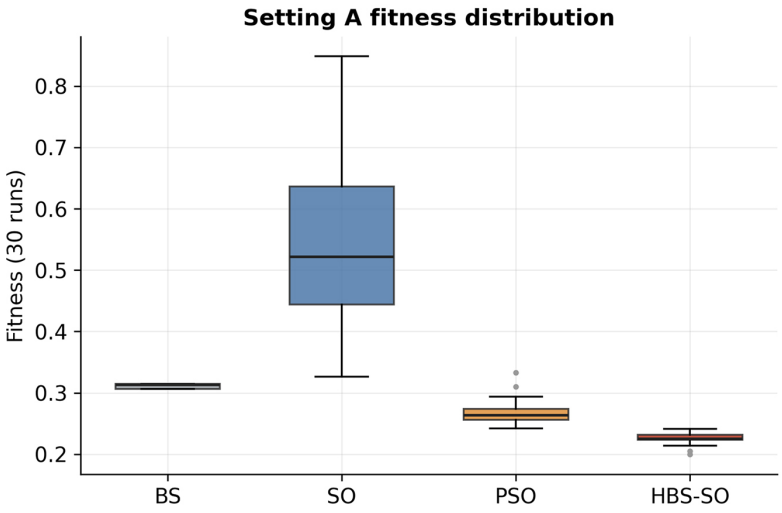


Fig. 7. Setting A fitness distribution over 30 runs

Note: HBS-SO combines the lowest median fitness with a tight interquartile range.

6.2 Setting B (multi-resource, 4N)

In the more demanding multi-resource formulation, HBS-SO again ranks first of four on weighted fitness (0.1443), significantly ahead of BS (+47.1%, $p < 0.001$) and the SO (+14.5%, $p < 0.001$) and marginally ahead of the strongest baseline, PSO (+0.9%). The difference from PSO is not statistically significant ($p = 0.40$), so on the weighted objective the two methods are best-in-class and statistically on par as shown in Table 5, and Table 6. What separates HBS-SO is that it holds this top position while also delivering the lowest energy consumption (714 J, marginally below PSO's 717 J and well below the 1075–1513 J of SO and BS) as shown in Figure 10, the lowest worst-case latency (14.4 s versus 15.2 s for PSO), and the highest deadline-satisfaction rate (99.8%) as shown in Figure 9. The enlarged 4N decision space rewards free exploration, which is why a well-tuned PSO is competitive here; the beam-seeded, mimetically-refined hybrid matches it on the objective and edges ahead on the operational metrics that matter for deployment. As illustrated in Figure 8.

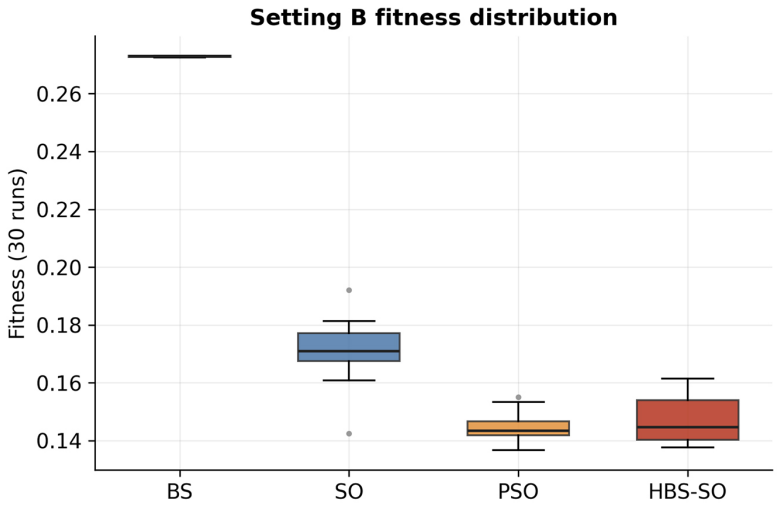


Fig. 8. Setting B fitness distribution over 30 runs

Note: HBS-SO and PSO are statistically on par and lead the other baselines.

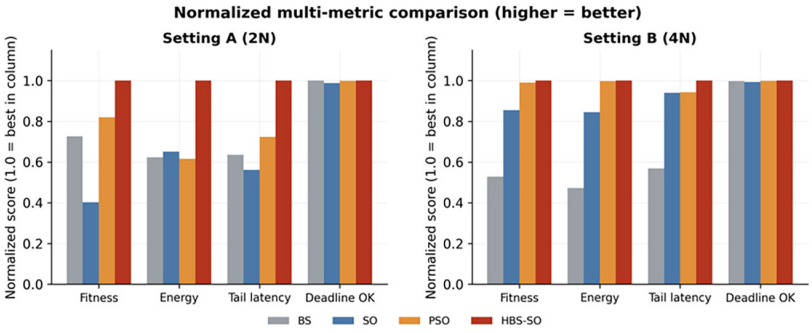


Fig. 9. Normalized multi-metric comparison

6.3 Cross-setting analysis

Comparing the two formulations directly confirms that HBS-SO is the strongest overall method: it ranks first of four on weighted fitness in both settings as shown in Table 7. In the offloading-centric formulation, the lead is decisive and broad—HBS-SO is best on weighted fitness (significantly ahead of every baseline, including PSO), energy, worst-case latency, deadline satisfaction, and run-to-run stability. In the multi-resource formulation, the objective becomes harder because the enlarged continuous space rewards free exploration; there, HBS-SO is statistically on par with the best baseline (PSO) on weighted fitness while still delivering the lowest energy, the lowest worst-case latency and the highest deadline-satisfaction rate. The consistent pattern across both regimes is that the combination of structural BS, population-based exploration and memetic refinement yields an optimizer that is either the outright best or tied-best on the objective, and best on the operational metrics—energy, tail latency and deadline satisfaction—that govern real MEC deployments.

6.4 Scalability and sensitivity

Across problem sizes from 50 to 400 tasks, HBS-SO's fitness advantage over BS not only persists but grows steadily with scale—from 21.4% at N = 50 to 43.9% at N = 400 (refer to Table 8, see Figure 11). This is an encouraging property: as the offloading problem becomes larger and harder, the value of combining structured search with memetic refinement over a plain heuristic increase rather than erodes, indicating that HBS-SO is well suited to the hundreds-of-tasks regime encountered in practice.

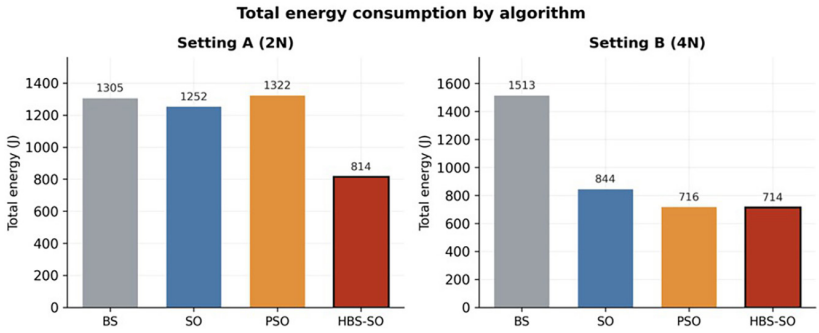


Fig. 10. Total energy consumption across algorithms

Sensitivity analysis on the multi-resource formulation shows stable, well-behaved tuning. Increasing the beam width improves fitness modestly and then

saturates—from 0.154 at width 4 to 0.144 at width 12, with width 16 giving no further gain—so a beam width of 10–12 is a sound default. Population size shows the expected trend, with the best result at a population of 30 (0.141); values of 20–30 is all reasonable (refer to Table 9, see Figure 12). The framework is therefore robust to its two main hyperparameters over practical ranges.

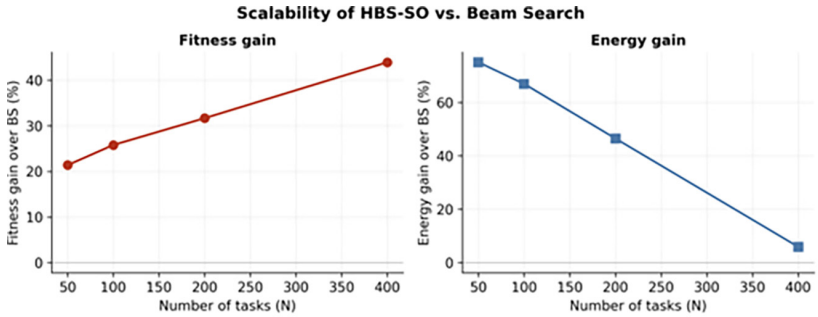


Fig. 11. Fitness and energy gain of HBS-SO over beam search versus problem size; the fitness advantage grows monotonically with scale

Table 5. Setting B multi-metric results over 30 runs

Algorithm	Fitness (mean ± std)	Fitness (best)	Energy (J)	Lat. Mean (s)	Lat. Max (s)	Deadline OK
BS	0.2729 ± 0.0002	0.2725	1513.3	2.85	25.21	99.5
SO	0.1687 ± 0.0093	0.1425	844.4	4.79	15.28	99.2
PSO	0.1456 ± 0.0066	0.1345	716.5	5.09	15.23	99.7
HBS-SO	**0.1443 ± 0.0101**	**0.1296**	**714.3**	4.92	**14.36**	**99.8**

Note: Lower fitness, energy, and latency values indicate better performance.

Table 6. Setting B Wilcoxon signed-rank tests (HBS-SO versus each baseline)

Comparison	W	*p*-value	Significant	Δ Mean (%)
HBS-SO vs. BS	0.00	0.00000	Yes	47.13
HBS-SO vs. SO	7.00	0.00000	Yes	14.48
HBS-SO vs. PSO	191.00	0.40449	No	0.92

Note: The PSO comparison is not statistically significant ($p = 0.40$), indicating a statistical tie.

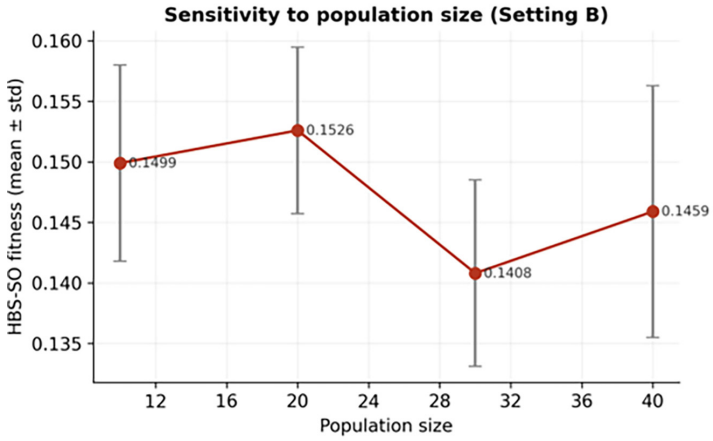


Fig. 12. Sensitivity of HBS-SO to population size (Setting B)

Table 7. Cross-setting of HBS-SO behavior across the two formulations

Metric	Setting A (2N)	Setting B (4N)
Best fitness (HBS-SO)	0.1998	0.1296
HBS-SO fitness rank	#1 of 4	#1 of 4
HBS-SO energy (J)	814.5	714.3
BS energy (J)	1305.3	1513.3

Table 8. Scalability of HBS-SO versus beam search across different problem sizes (Setting A)

N	BS Fitness	HBS-SO Fitness	Fitness Gain (%)	Energy Gain (%)
50	0.1280	0.1006	21.38	75.08
100	0.1659	0.1231	25.79	66.99
200	0.3117	0.2130	31.66	46.51
400	0.6194	0.3474	43.91	5.94

Table 9. Sensitivity analysis of HBS-SO (Setting B) with respect to beam width and population size (mean ± std fitness)

Parameter	Value	Mean Fitness	Std
Beam width	4	0.1544	0.0051
Beam width	8	0.1498	0.0083
Beam width	12	0.1440	0.0146
Beam width	16	0.1445	0.0140
Population	10	0.1499	0.0081
Population	20	0.1526	0.0069
Population	30	0.1408	0.0077
Population	40	0.1459	0.0104

7 LIMITATIONS

- In the multi-resource formulation, HBS-SO's fitness advantage over PSO is marginal and not statistically significant; on the weighted objective, the two methods are best-in-class and statistically on par, with HBS-SO's clearer advantages appearing on energy, worst-case latency, and deadline satisfaction.
- Evaluation is simulation-based; the CMOS energy model and Shannon-rate transmission are analytical approximations, and physical-device validation is left to future work.
- The multi-objective weight vectors are fixed; different weightings could alter the relative rankings, and a full weight-sensitivity study is beyond the present scope.
- The study models a single user; multi-user contention, mobility, and hand-off are not considered.
- The present evaluation fixes the memetic local-search budget; a systematic study of the trade-off between local-search effort and solution quality is left to future work.

8 CONCLUSION

We presented HBS-SO, an intelligent resource-optimization framework that integrates BS, the SO and a memetic local-search stage for task offloading and resource allocation in MEC. Evaluated on two problem formulations BS, the SO and PSO over 30 independent runs, HBS-SO ranks first on weighted fitness in both settings. In the offloading-centric formulation the lead is decisive: it significantly outperforms every baseline including PSO (+18.0%, $p < 0.001$) and at the same time delivers the lowest energy (a 38% reduction), the lowest worst-case latency (a 27–37% reduction) and full deadline satisfaction, while being the most stable optimizer. In the harder multi-resource formulation, it remains first on mean fitness but is statistically level with PSO on the objective; its clearer advantages there are the lowest energy, the lowest worst-case latency and the highest deadline satisfaction rate. The practical takeaway is that pairing a structural initializer with population search and a targeted local-refinement stage yield an optimizer that is competitive-to-leading on the objective and consistently strong on the metrics operators actually provision for. Future work will pursue physical-device validation, multi-user contention and mobility models, ablation of the individual stages under a common evaluation budget, comparison against additional recent metaheuristic and learning-based baselines, and adaptive objective weighting.

9 REFERENCES

[1] L. Ale, N. Zhang, X. Fang, X. Chen, S. Wu, and L. Li, "Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 3, pp. 881–892, 2021. https://doi.org/10.1109/TCCN.2021.3066619

[2] M. Babar, M. S. Khan, A. Din, F. Ali, U. Habib, and K. S. Kwak, "Intelligent computation offloading for IoT applications in scalable edge computing using artificial bee colony optimization," *Complexity*, vol. 2021, no. 1, Art. no. 5563531, 2021. https://doi.org/10.1155/2021/5563531

[3] J. Bi, K. Zhang, H. Yuan, and J. Zhang, "Energy-efficient computation offloading for static and dynamic applications in hybrid mobile edge cloud system," *IEEE Transactions on Sustainable Computing*, vol. 8, no. 2, pp. 232–244, 2022. https://doi.org/10.1109/TSUSC.2022.3216461

[4] Q. You and B. Tang, "Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things," *Journal of Cloud Computing*, vol. 10, no. 1, Art. no. 41, 2021. https://doi.org/10.1186/s13677-021-00256-4

[5] S. Li, H. Ge, X. Chen, L. Liu, H. Gong, and R. Tang, "Computation offloading strategy for improved particle swarm optimization in mobile edge computing," in *Proc. 2021 IEEE 6th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA),* 2021, pp. 375–381. https://doi.org/10.1109/ICCCBDA51879.2021.9442609

[6] H. Zhao, Y. Liu, X. Liu, Y. Xie, and P. Wu, "Enhanced task offloading in mobile edge computing: A hybrid approach using deep Q-learning from demonstrations and heuristic optimization," in *Proc. 2025 International Conference on Electrical Automation and Artificial Intelligence (ICEAAI),* 2025, pp. 1321–1324. https://doi.org/10.1109/ICEAAI64185.2025.10956500

[7] C. Zhang, Y. Deng, H. Zhao, T. Chen, and S. Deng, "Tail-learning: Adaptive learning method for mitigating tail latency in autonomous edge systems," *ACM Transactions on Autonomous and Adaptive Systems,* vol. 20, no. 4, pp. 1–25, 2025. https://doi.org/10.1145/3737289

[8] F. A. Hashim and A. G. Hussien, "Snake optimizer: A novel meta-heuristic optimization algorithm," *Knowledge-Based Systems*, vol. 242, Art. no. 108320, 2022. https://doi.org/10.1016/j.knosys.2022.108320

[9] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, 2020. https://doi.org/10.1109/JIOT.2020.3024223

[10] B. Rong, "Energy efficient computation offloading in mobile edge computing," *IEEE Wireless Communications,* vol. 30, no. 2, pp. 8–8, 2023. https://doi.org/10.1109/MWC.2023.10105148

[11] S. Dong, Y. Xia, and J. Kamruzzaman, "Quantum particle swarm optimization for task offloading in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 8, pp. 9113–9122, 2022. https://doi.org/10.1109/TII.2022.3225313

[12] M. A. E. Rasool, A. Kumar, and A. Islam, "Dynamic task offloading optimization in mobile edge computing systems with time-varying workloads using improved particle swarm optimization," *International Journal of Advanced Computer Science & Applications*, vol. 15, no. 4, 2024. https://doi.org/10.14569/IJACSA.2024.01504122

[13] H. Zhou, Z. Zhang, Y. Wu, M. Dong, and V. C. Leung, "Energy efficient joint computation offloading and service caching for mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 2, pp. 950–961, 2022. https://doi.org/10.1109/TGCN.2022.3186403

[14] S. C. Palaniappan and P. P. Ponnuswamy, "Task offloading in edge computing using integrated particle swarm optimization and genetic algorithm," *Advances in Science and Technology Research Journal*, vol. 19, no. 1, pp. 371–380, 2025. https://doi.org/10.12913/22998624/195658

[15] S. Ala'A and A. Y. Alma'aitah, "Distributed task offloading in mobile edge computing using meta-heuristics," in *Proc. 2024 6th International Conference on Communications, Signal Processing, and Their Applications (ICCSPA),* 2024, pp. 1–6. https://doi.org/10.1109/ICCSPA61559.2024.10794208

[16] J. Mei, Z. Tong, K. Li, L. Zhang, and K. Li, "Energy-efficient heuristic computation offloading with delay constraints in mobile edge computing," *IEEE Transactions on Services Computing,* vol. 16, no. 6, pp. 4404–4417, 2023. https://doi.org/10.1109/TSC.2023.3324604

[17] A. Zabi, "Ajegwo: A hybrid metaheuristic for joint task offloading and resource management in MEC," *International Journal of Applied Mathematics*, vol. 38, no. 6s, pp. 799–816, 2025. https://doi.org/10.12732/ijam.v38i6s.435

[18] S. H. Shakir, A. A. Ghani, A. N. Kadhum, A. N. Kadhim, and A. N. Kadhim, "Hybrid swarm-intelligence framework for TSP optimization and energy-aware MEC offloading: Accelerated convergence and resource efficiency," in *Proc. 2025 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob),* 2025, pp. 31–37. https://doi.org/10.1109/APWiMob67231.2025.11269124

[19] A. Nain, S. Sheikh, R. Malik, and S. U. Mushtaq, "Hybrid metaheuristic based resource allocation approach in SDN-based edge computing environment," in *Proc. 2023 2nd International Conference on Futuristic Technologies (INCOFT),* 2023, pp. 1–5. https://doi.org/10.1109/INCOFT60753.2023.10425524

[20] T. Zhang and X. Guo, "Efficient task offloading using ant colony optimization and reptile search algorithms in edge computing for things context," *International Journal of Advanced Computer Science & Applications,* vol. 15, no. 9, pp. 588–596, 2024. https://doi.org/10.14569/IJACSA.2024.0150960

[21] P. Ramakrishnan, C. H. Basha, B. T. Selvi, K. V, M. Saravanan, and R. Ramya. "Energy-efficient resource allocation in 5G massive MIMO networks using hybrid metaheuristic optimization," in *Proc. 2025 2nd International Conference on Electronic Circuits and Signaling Technologies (ICECST),* 2025, pp. 401–408. https://doi.org/10.1109/ICECST66106.2025.11307213

[22] M. F. Dehkordi and B. Jabbari, "Efficient and sustainable task offloading in UAV-assisted MEC systems via meta deep reinforcement learning," in *Proc. ICC 2025—IEEE International Conference on Communications,* 2025, pp. 1353–1358. https://doi.org/10.1109/ICC52391.2025.11161525

[23] M. Elkawkagy, I. A. Elgendy, S. A. Chelloug, and H. Elbeh, "Genetic algorithm-driven joint optimization of task offloading and resource allocation for fairness-aware latency minimization in mobile edge computing," *IEEE Access*, vol. 13, pp. 118237–118248, 2025. https://doi.org/10.1109/ACCESS.2025.3584971

[24] S. Fu, X. B. Zhai, C. Yi, L. Pang, and C. W. Tan, "Energy efficient and low latency computation offloading via truly PPO in mobile edge computing networks with multi-UAV," in *Proc. 2024 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics,* 2024, pp. 98–104. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics62450.2024.00038

[25] M. Zulfiqar and A. Ahsan, "Deadline-aware DRL for MEC task scheduling: Balancing offloading and deferral to maximize completion rates," in *Proc. 2026 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA),* 2026, pp. 1–6. https://doi.org/10.1109/ACDSA67686.2026.11467863

[26] H. Xiao, Z. Hu, X. Zhang, A. Xu, M. Zheng, and K. Li, "Federated deep reinforcement learning for task offloading in MEC-enabled heterogeneous networks," *IEEE Internet of Things Journal,* vol. 12, no. 8, pp. 10238–10252, 2024. https://doi.org/10.1109/JIOT.2024.3509893

[27] C. Silva, N. Magaia, and A. Grilo, "Task offloading optimization in mobile edge computing based on deep reinforcement learning," in *Proc. International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2023, pp. 109–118. https://doi.org/10.1145/3616388.3617539

[28] L. Dai, J. Mei, Z. Yang, Z. Tong, C. Zeng, and K. Li, "Lyapunov-guided deep reinforcement learning for delay-aware online task offloading in MEC systems," *Journal of Systems Architecture,* vol. 153, Art. no. 103194, 2024. https://doi.org/10.1016/j.sysarc.2024.103194

[29] H. Zhou, K. Jiang, X. Liu, X. Li, and V. C. Leung, "Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing," *IEEE Internet of Things Journal,* vol. 9, no. 2, pp. 1517–1530, 2021. https://doi.org/10.1109/JIOT.2021.3091142

[30] Y. Qin, J. Chen, L. Jin, R. Yao, and Z. Gong, "Task offloading optimization in mobile edge computing based on a deep reinforcement learning algorithm using density clustering and ensemble learning," *Scientific Reports*, vol. 15, no. 1, Art. no. 211, 2025. https://doi.org/10.1038/s41598-024-84038-3

[31] W. Fan, F. Xiao, Y. Pan, X. Chen, L. Han, and S. Yu, "Latency-aware joint task offloading and energy control for cooperative mobile edge computing," *IEEE Transactions on Services Computing*, vol. 18, no. 3, pp. 1515–1528, 2025. https://doi.org/10.1109/TSC.2025.3553708

[32] P. Liu, J. Lei, and W. Liu, "Learning-based latency and energy optimization in SCMA-enhanced UAV-MEC networks," in *Proc. 2023 IEEE/CIC International Conference on Communications in China (ICCC),* 2023, pp. 1–6. https://doi.org/10.1109/ICCC57788.2023.10233490

[33] M. Paknejad, P. F. Moshiri, M. Simsek, B. Kantarci, and H. T. Mouftah, "A reliable and efficient 5G vehicular MEC: Guaranteed task completion with minimal latency," in *Proc. 2025 IEEE International Conference on Communications Workshops (ICC Workshops*), 2025, pp. 566–571. https://doi.org/10.1109/ICCWorkshops67674.2025.11162463

[34] R. Dasari and S. K. Nayak, "Reto: Energy-efficient and reliable task offloading scheme with deadline in fog-enabled IoT networks," in *Proc. 2025 17th International Conference on Communication Systems and Networks (COMSNETS),* 2025, pp. 275–283. https://doi.org/10.1109/COMSNETS63942.2025.10885694

[35] B. Sonwani and H. A. Marquering, "Delay-sensitive task offloading for emergency ambulance in edge-MEC-cloud continuum using deep reinforcement learning," in *Proc. 2025 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN),* 2025. https://doi.org/10.1109/NFV-SDN66355.2025.11349471

[36] D. K. Nishad, V. R. Verma, P. Rajput, S. Gupta, A. Dwivedi, and D. R. Shah, "Adaptive AI-enhanced computation offloading with machine learning for QoE optimization and energy-efficient mobile edge systems," *Scientific Reports*, vol. 15, Art. no. 15263, 2025. https://doi.org/10.1038/s41598-025-00409-4

[37] H. Asadian-Rad, H. Soleimani, and S. Farahmand, "Energy efficient task offloading in UAV-enabled MEC using a fully decentralized deep reinforcement learning approach," *arXiv preprint arXiv:2508.06863,* 2025. https://doi.org/0.48550/arXiv.2508.06863.

[38] J. Erskine and D. V. Huynh, "Joint computation offloading and resource allocation for mobile edge computing," in *International Conference on Industrial Networks and Intelligent Systems*, in Lecture Notes in Computer Science, Cham, Switzerland: Springer Nature, 2023, pp. 51–62. https://doi.org/10.1007/978-3-031-47359-3_4

[39] X. Zhang, X. Zhang, and W. Yang, "Joint offloading and resource allocation using deep reinforcement learning in mobile edge computing," *IEEE Transactions on Network Science and Engineering,* vol. 9, no. 5, pp. 3454–3466, 2022. https://doi.org/10.1109/TNSE.2022.3184642

[40] J. Bi, Z. Sun, H. Yuan, and J. Zhang, "Joint optimization of cache-assisted offloading and resource allocation in mobile edge computing," in *Proc. 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC),* 2023, pp. 4997–5002. https://doi.org/10.1109/SMC53992.2023.10394647

[41] G. Pradeep, S. Ramamoorthy, M. Krishnamurthy, P. S. Rajakumar, and V. Saritha, "Hybrid energy-efficient task offloading algorithm (HEETA): A framework for optimizing edge computing offloading decisions," *Journal of Electrical Systems*, vol. 20, no. 5s, pp. 92–111, 2024. https://doi.org/10.52783/jes.1835

[42] C. Eang *et al.*, "Offloading decision and resource allocation in mobile edge computing for cost and latency efficiencies in real-time IoT," *Electronics*, vol. 13, no. 7, Art. no. 1218, 2024. https://doi.org/10.3390/electronics13071218

[43] J. Li, H. Yuan, D. Guo, J. Bi, and J. Zhang, "Intelligent offloading of dependent tasks for UAV-relaying-assisted mobile edge computing," in *Proc. 2025 IEEE International Conference on Systems, Man, and Cybernetics (SMC),* 2025, pp. 2852–2857. https://doi.org/10.1109/SMC58881.2025.11343136

[44] N. Nujhat *et al.*, "Task offloading exploiting grey wolf optimization in collaborative edge computing," *Journal of Cloud Computing*, vol. 13, no. 1, Art. no. 23, 2024. https://doi.org/10.1186/s13677-023-00570-z

10 AUTHORS

Esra'a Alhenawi is an accomplished academic and researcher in the field of Computer Science, currently serving as a Full-time lecturer at Department of

Information Technology, Faculty of Prince Al-Hussein Bin Abdallah II for Information Technology, Al al-Bayt University. Dr. Alhenawi earned a PhD in Computer Science from the University of Jordan (2017–2022), graduating with an excellent GPA. Their research interests include cybersecurity in the Internet of Things (IoT), cloud computing, mobile ad hoc networks (MANETs), Machine learning, and network security. They have contributed to several peer-reviewed journals and publications. With a strong commitment to innovation, education, and research, she continues to make significant contributions to the field of computer science, particularly in applying intelligent systems to real-world challenges (E-mail: esraahenawi@aabu.edu.jo).

Omar Alidmat is with the Department of Computer Science, Faculty of Information Technology, Zarqa University, Zarqa, Jordan. He received his B.Sc. degree in Computer Information Systems, M.Sc. degree in Computer Science from Al al-Bayt University, Jordan, and the Ph.D. degree in Computer Science from Universiti Sains Malaysia, Malaysia. His research interests also include modelling the decision-making processes of evacuees in terms of their movements, judgments on surrounding conditions, and choices of actions (e.g., stepping back, moving ahead, backtracking, switching lanes, waiting) during fire crowd evacuation. His work also accounts for crowd pressure and behaviors (e.g., arching, clogging, competitiveness, falling, trampling, crushing, overtaking, and rushing simultaneously towards exits) during fire evacuations, particularly in overcrowded scenarios (E-mail: oalidmat@zu.edu.jo).

Esraa Abu Elsoud received the B.Sc. degree in Electrical Engineering from The Hashemite University, Jordan, in 2013, and the M.Sc. degree in Cyber Security from The Hashemite University in 2023. She works as a lecturer at Cybersecurity and Cloud Computing Department, Faculty of Information Technology, Applied Science Private University in the fields of cybersecurity and cloud computing. Her current research interests include cybersecurity, machine learning, big data, and mobile networks (E-mail: e_abuelsoud@asu.edu.jo).