

QoE-Driven Task Offloading under Shared Resources in 5G-Enabled IoT Systems

Kadjo Tanon Lambert^{1*}, Kone Kigninman Désiré², Bitá Romaric de Judidael³, Asseu Olivier^{1,2}

¹Institut National Polytechnique Houphouët-Boigny (INP-HB), Yamoussoukro, Côte d'Ivoire

²École Supérieure Africaine des TIC (ESATIC), Abidjan, Côte d'Ivoire

³Institut Pédagogique National de l'Enseignement Technique et Professionnel (IPNETP), Abidjan, Côte d'Ivoire

Email: *tanon.kadjo@inphb.ci

How to cite this paper: Lambert, K.T., Désiré, K.K., de Judidael, B.R. and Olivier, A. (2026) QoE-Driven Task Offloading under Shared Resources in 5G-Enabled IoT Systems. *Journal of Sensor Technology*, 16, 81-96.

<https://doi.org/10.4236/jst.2026.164006>

Received: September 15, 2026

Accepted: October 24, 2026

Published: October 27, 2026

Copyright © 2026 by author(s) and Scientific Research Publishing Inc. This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

Abstract

Computation-intensive Internet of Things (IoT) applications require offloading decisions that account for latency, device-side energy consumption, and contention for shared edge resources. This study proposes a hybrid metaheuristic for ternary task assignment across local devices, Mobile Edge Computing (MEC), and cloud resources. The offloading model couples user decisions through shared uplink bandwidth and MEC computing capacity. Magnificent Frigatebird Optimization (MFO) provides the main population-search mechanism, while Piranha Foraging Optimization Algorithm (PFOA) contributes a selectively accepted nonlinear reverse-evasion refinement around the incumbent best solution. The proposed approach is evaluated on 400-user workloads over 30 paired random seeds using a matched search budget of 11,250 fitness evaluations against MFO, HGSTO, and discrete Particle Swarm Optimization (PSO). It achieves a Quality of Experience (QoE) satisfaction ratio of 0.5769, a mean latency of 0.4926 s, and a mean device-side energy consumption of 0.0933 J, compared with QoE values of 0.5719 for MFO and 0.5242 for HGSTO. The QoE improvement over MFO remains statistically significant under the paired t-test, Wilcoxon signed-rank test, and Holm correction. In addition, exhaustive evaluation on ten tractable 9-task instances shows that the proposed method reaches the exact optimum in all cases. Overall, the results indicate that selective PFOA refinement provides a small but consistent improvement over the MFO backbone under the considered resource-coupled offloading scenario.

Keywords

5G IoT, Computation Offloading, Mobile Edge Computing, PiFBOA, Metaheuristic Optimization, Shared Resources, QoE, Latency

1. Introduction

The integration of fifth generation (5G) networks with the Internet of Things (IoT) supports smart-city, industrial, healthcare, and connected-mobility applications [1]-[8]. Despite the communication capabilities of 5G, many IoT devices remain constrained by limited battery capacity and processing resources. Local execution of computation-intensive tasks can therefore increase delay and device energy use, whereas remote execution introduces transmission overhead [9]-[15].

Computation offloading addresses this trade-off by assigning tasks to local processors, nearby MEC servers, or remote cloud resources [16]. The decision is complicated by variations in task size, computational demand, channel conditions, deadlines, and available resources. It is also coupled across users when several tasks compete for the same radio bandwidth or MEC processing capacity [14] [17]-[19]. Under these conditions, selecting an execution location for each task becomes a combinatorial resource-allocation problem rather than a collection of independent decisions.

Deep reinforcement learning and federated learning have been applied to adaptive offloading [5] [20]-[23]. They can represent complex decision environments but require a training process and explicit state-action design. Metaheuristic methods offer a training-free alternative, although their effectiveness depends on the balance between global exploration and local refinement [24]-[27].

This study develops PiFBOA, a hybrid MFO-PFOA optimizer for Device-MEC-Cloud task offloading under shared uplink and MEC resources. The novelty is not claimed to arise from a new elementary search operator. Instead, MFO is preserved as the identifiable population-search backbone, while PFOA-derived nonlinear control and reverse-evasion are restricted to a separate best-solution refinement stage. A refinement is accepted only when it improves the score after ternary decoding, which makes the contribution of the PFOA component directly testable against MFO alone.

The contributions are: 1) a resource-coupled Device-MEC-Cloud formulation with shared uplink bandwidth and MEC CPU capacity; 2) an explicit MFO-PFOA hybridization in which MFO performs the population-level search and PFOA-derived nonlinear reverse-evasion generates selectively accepted, peer-guided refinements of the incumbent best solution after ternary decoding; 3) a parameter-selection protocol using validation seeds disjoint from the final test seeds; 4) a matched-fitness-evaluation comparison with MFO, adapted HGSTO, discrete PSO, and a deterministic greedy heuristic; and 5) a small-scale exhaustive validation against the exact optimum.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 presents the system model and objective. Section 4 describes PiFBOA. Section 5 presents the experimental methodology. Section 6 reports and discusses the results, and Section 7 concludes the paper.

2. Related Work and Research Gap

Task offloading has been widely studied using game-theoretic optimization, rein-

forcement learning, scheduling, and population-based search. Keshta *et al.* [28] investigated game-theoretic optimization for IoT task offloading, while Cumia *et al.* [2] used distributed multi-agent Q-learning to reduce energy consumption at the extreme edge. Eang *et al.* [19] jointly considered offloading decisions and resource allocation for real-time IoT applications, highlighting the importance of accounting for both communication and computing constraints.

Learning-based approaches have also received considerable attention. M2FD combines federated learning and deep reinforcement learning for multi-type task offloading [5]. Wang *et al.* [21] addressed dependent-task offloading using deep reinforcement learning, while Nieto *et al.* [23] considered dynamic task offloading across the 5G edge-cloud continuum. These methods can adapt to complex operating conditions, but they rely on model training and on the definition of suitable state and action spaces. M2FD is therefore retained as an important learning-based reference in this study, although it is not used as an experimental baseline because a source-faithful implementation of its federated learning pipeline was not available.

Metaheuristic approaches provide an alternative that does not require policy training. Ben Sada *et al.* [24] proposed HGSTO, which combines genetic search and simulated annealing for selective inference-task offloading under time and energy constraints. Chen *et al.* [25] introduced a multi-population quantum-behavior particle swarm approach for IoT task offloading. At the optimizer level, PFOA [26] already includes nonlinear control and diversity-preserving mechanisms, whereas MFO [27] combines exploration toward better population members with exploitation around the current best solution.

The gap considered in this work is therefore not the introduction of a new elementary search operator. Rather, it concerns the controlled integration of complementary search mechanisms within a resource-coupled Device-MEC-Cloud offloading problem. Existing studies either focus on learning-based policies, apply metaheuristics directly to offloading, or combine multiple search operators without explicitly isolating the contribution of a selective refinement stage under shared uplink and MEC constraints. This study addresses that gap by preserving MFO as the population-search backbone and introducing a PFOA-derived reverse-evasion refinement only around the incumbent best solution, with acceptance conditioned on an improvement after ternary decoding. This design allows the effect of the refinement stage to be evaluated directly against an MFO-only baseline under a matched fitness-evaluation budget.

3. System Model and Problem Formulation

The considered system consists of three execution layers: IoT devices, Mobile Edge Computing (MEC) resources connected to the access network, and remote cloud resources [19] [23]. Each computational task is assigned to exactly one of these layers. Offloaded tasks share the available uplink bandwidth, while tasks assigned to the MEC layer also share the available MEC computing capacity. Con-

sequently, the performance of one task depends not only on its own characteristics but also on the offloading decisions made for the other tasks.

3.1. Task Representation and Decision Variable

Consider a set of N computational tasks. Task i is characterized by

$$(D_i, C_i, \tau_i, E_i^{\max}),$$

where D_i denotes the input data size, C_i the required number of CPU cycles, τ_i the maximum tolerated latency, and $E_i^{\max}$ the maximum device-side energy allowed for the task.

The execution decision for task i is represented by the ternary variable

$$a_i \in \{0, 1, 2\}, i = 1, \dots, N \quad (1)$$

where $a_i = 0$ denotes local execution, $a_i = 1$ MEC execution, and $a_i = 2$ cloud execution. The complete offloading solution is denoted by

$$A = [a_1, a_2, \dots, a_N].$$

3.2. Local, MEC, and Cloud Execution

For local execution, let f_i denote the CPU frequency of device i , and let κ be the effective switching-capacitance coefficient. The corresponding execution latency and energy consumption are

$$T_i^{\text{loc}} = \frac{C_i}{f_i} \quad (2)$$

and

$$E_i^{\text{loc}} = \kappa C_i f_i^2. \quad (3)$$

For offloaded tasks, the total uplink bandwidth B is shared equally among all tasks assigned to either MEC or cloud. Let $N_{\text{off}}(A)$ denote the number of offloaded tasks under assignment A . The bandwidth allocated to each offloaded task is therefore

$$B_i(A) = \frac{B}{\max(N_{\text{off}}(A), 1)}. \quad (4)$$

The corresponding uplink transmission rate is

$$R_i(A) = B_i(A) \log_2 \left(1 + \frac{P_i h_i}{N_0 B_i(A)} \right), \quad (5)$$

where P_i is the transmit power of device i , h_i is the channel power gain, and N_0 is the noise power spectral density. The uplink transmission time is then

$$T_i^{\text{up}}(A) = \frac{D_i}{R_i(A)}. \quad (6)$$

For MEC execution, let F_{MEC} denote the aggregate MEC computing capacity and $N_{\text{MEC}}(A)$ the number of tasks assigned to the MEC layer. Equal CPU shar-

ing is assumed, giving

$$f_i^{MEC}(A) = \frac{F_{MEC}}{\max(N_{MEC}(A), 1)}. \quad (7)$$

The MEC execution latency and device-side energy consumption are

$$T_i^{MEC}(A) = T_i^{up}(A) + \frac{C_i}{f_i^{MEC}(A)} \quad (8)$$

and

$$E_i^{MEC}(A) = P_i T_i^{up}(A). \quad (9)$$

For cloud execution, a fixed backhaul delay T_{bh} and an effective per-task cloud computing rate f_{cloud} are considered. The cloud CPU rate is not shared among cloud-assigned tasks; therefore, cloud-side CPU contention is not explicitly modeled. The resulting latency is

$$T_i^{cloud}(A) = T_i^{up}(A) + T_{bh} + \frac{C_i}{f_{cloud}}, \quad (10)$$

While the corresponding device-side energy consumption is

$$E_i^{cloud}(A) = P_i T_i^{up}(A). \quad (11)$$

The result-download time is neglected because the returned output is assumed to be small relative to the uploaded task data. Energy consumption is evaluated from the IoT-device perspective; MEC and cloud infrastructure energy are, therefore, outside the scope of the present model. Future work will extend the model to include downlink transmission delay and infrastructure-side energy consumption at the MEC and cloud layers.

A key consequence of this formulation is that offloading decisions are mutually dependent. Changing the execution layer of one task may alter both the bandwidth available to all offloaded tasks and the computing capacity available to MEC-assigned tasks.

3.3. QoE and Search Objective

For a given assignment A , the latency $T_i(A)$ and energy $E_i(A)$ of task i correspond only to the execution layer selected by a_i . A task is considered successfully served when it satisfies both its latency and device-side energy constraints:

$$q_i(A) = 1 \left[T_i(A) \leq \tau_i \wedge E_i(A) \leq E_i^{\max} \right]. \quad (12)$$

The Quality of Experience (QoE) satisfaction ratio is then defined as

$$\text{QoE}(A) = \frac{1}{N} \sum_{i=1}^N q_i(A). \quad (13)$$

Because QoE is discrete, several different assignments may obtain the same satisfaction ratio. To distinguish such solutions during the search, a secondary normalized latency-energy term is introduced:

$$\Psi(A) = \frac{1}{2} \text{mean} \left[\min \left(\frac{T_i(A)}{\tau_i}, 10 \right) \right] + \frac{1}{2} \text{mean} \left[\min \left(\frac{E_i(A)}{E_i^{\max}}, 10 \right) \right]. \quad (14)$$

The optimization score is defined as

$$S(A) = \text{QoE}(A) - \eta \Psi(A), \quad (15)$$

where η controls the contribution of the secondary normalized latency-energy term to the scalar optimization score. For $\eta > 0$, candidate assignments are ranked according to the combined score S , and an assignment with slightly lower QoE may therefore be preferred if the latency-energy contribution is sufficiently favorable. The sensitivity of the results to η is examined in Section 5.2. In the final configuration selected on the disjoint validation seeds, $\eta = 0$; consequently, the final optimization runs rank candidate assignments according to QoE alone. Mean latency and mean device-side energy are reported separately as performance metrics.

4. Proposed Method

This section describes the proposed PiFBOA method.

4.1. Ternary Decoding and MFO Backbone

MFO and PFOA operate in continuous search spaces, whereas the offloading problem considered here is ternary. To avoid introducing an implicit initialization bias toward any execution layer, each candidate solution is represented by a continuous vector $X_r = [x_{r,1}, \dots, x_{r,N}]$, with $x_{r,j} \in [0, 3)$, and decoded only when its fitness is evaluated:

$$a_j = \min(2, \lfloor x_{r,j} \rfloor). \quad (16)$$

Accordingly, the intervals $[0, 1)$, $[1, 2)$, and $[2, 3)$ correspond respectively to local, MEC, and cloud execution. Since the three intervals have equal widths, uniform initialization over $[0, 3)$ gives the three execution layers equal initial selection probability.

The MFO backbone follows the search structure reported in [27]. Candidate positions are initialized uniformly over $[0, 3)$. During exploration, each candidate is guided by a randomly selected population member with a better score; when no better member exists, the current global best is used instead. The resulting candidate is decoded and accepted only if its optimization score does not decrease.

The exploration update is

$$x_{r,j}^M = x_{r,j} + (1 - 2u_{r,j})(SS_{r,j} - I_{r,j}x_{r,j}), \quad (17)$$

where $u_{r,j} \sim U(0, 1)$, $I_{r,j} \in \{1, 2\}$, and $SS_{r,j}$ denotes the j -th component of the selected guiding solution.

During exploitation, the candidate moves toward the current global best according to

$$x_{r,j}^D = x_{r,j} + (1 - 2u_{r,j}) \frac{x_j^{best} - x_{r,j}}{t}, \quad (18)$$

where x_j^{best} is the j -th component of the current best solution and t is the iteration index. As the search progresses, the $1/t$ factor reduces the step magnitude and promotes progressively finer exploitation around the incumbent best solution.

4.2. Selective PFOA-Derived Refinement

The hybrid component introduces a separate refinement stage based on two mechanisms derived from PFOA [26]: nonlinear control and reverse-evasion direction. Rather than modifying the MFO population dynamics, this stage generates K peer-guided alternatives around the current best solution.

The nonlinear control factor is defined as

$$\alpha_t = \cos\left(\frac{\pi t}{2T}\right), \quad (19)$$

where T is the maximum number of iterations.

For each refinement trial, a peer solution X_r is selected from the population, and a component-wise random sign vector $s \in \{-1, +1\}^N$ is generated. The refined candidate is computed as

$$X^H = \text{clip}\left[X^{best} + \gamma \alpha_t s \odot (X_r - X^{best}), 0, 2\right], \quad (20)$$

where $\odot$ denotes element-wise multiplication and γ controls the refinement amplitude.

Each candidate generated by (20) is decoded using (16) and evaluated with the objective function defined in Section 3.3. The best refinement replaces the incumbent only if it strictly improves $S(A)$; otherwise, the current best solution is retained.

The term $X_r - X^{best}$ provides a search direction informed by the current population rather than by an independent random perturbation. The sign vector s can reverse this direction independently across dimensions, thereby increasing the diversity of the refinement candidates. At the same time, α_t decreases as the search progresses, so the refinement evolves from broader perturbations in early iterations toward a more localized search near convergence. The scale factor γ controls the overall strength of this mechanism and is selected using validation seeds that are separate from the final test set.

This design keeps the MFO search structure intact while using the PFOA-derived mechanism only as a selective refinement step. Its contribution can therefore be assessed directly through comparison with the MFO-only baseline. The complete procedure is summarized in **Algorithm 1**.

5. Experimental Methodology

5.1. Workload Generation and Simulation Settings

The main experiment considers 400 users, with one computational task generated

Algorithm 1. Proposed procedure.

Step	Procedure
1	Initialize the MFO population uniformly over $[0,3)^N$ and evaluate the assignments decoded using (16).
2	Apply MFO exploration using (17) and greedily accept non-degraded candidates.
3	Apply MFO exploitation using (18) and update the incumbent best solution.
4	Compute α_i using (19) and generate K PFOA-derived refinement candidates using (20).
5	Decode and evaluate the refinement candidates; replace the incumbent only if a strict improvement is achieved.
6	Repeat until the fitness-evaluation budget is exhausted, then return the decoded best assignment.

for each user. For each random seed, a single workload instance is generated and reused across all methods to ensure paired comparisons under identical conditions. Each workload instance is treated as a static scheduling snapshot in which all considered tasks are assumed to be available within the same scheduling interval. Dynamic task arrivals and departures during the interval are not modeled.

Unless otherwise stated, task computation demand is sampled from a discrete uniform distribution between 100 and 1000 million CPU cycles, and input data size from a discrete uniform distribution between 50 and 500 KiB. The latency requirement is sampled from $U(0.05, 1.0)$ s, the device-side energy requirement from $U(0.1, 5.0)$ J, and the local CPU frequency from $U(0.5, 2.0)$ GHz. Channel gain is sampled in dB from $U(-110, -60)$ and then converted to linear power gain. These distributions define a controlled synthetic workload and are not intended to reproduce measured application traces. The main simulation settings are summarized in **Table 1**.

Table 1. Main simulation settings.

Parameter	Value
Users/tasks	400/400
Population size	50
Reference iterations for parameter tuning	100
Matched NFE target for the final comparison	11,250
Total uplink bandwidth	10 MHz
Transmit power	0.1 W
Noise PSD	1.995×10^{-20} W/Hz
Aggregate MEC CPU capacity	100 GHz
Cloud CPU frequency	10 GHz
Cloud backhaul delay	0.020 s

Continued

Local energy coefficient κ	1×10^{-28}
Final $\gamma / K / \eta$	1.1/16/0
Final paired test seeds	30 (seeds 0 - 29)

5.2. Parameter Selection on Disjoint Validation Seeds

The algorithmic parameters were examined using eight validation seeds (200-207), which are completely disjoint from the 30 seeds used in the final evaluation. The scale factor γ was evaluated at 0.2, 0.5, 0.8, 1.1, and 1.4; the number of refinement trials K at 4, 8, 12, and 16; and the weight η of the secondary latency-energy term at 0, 0.01, 0.02, and 0.05.

Table 2 reports the mean QoE and mean latency obtained for each tested parameter value on the validation seeds. Among the tested values, $\gamma = 1.1$, $K = 16$, and $\eta = 0$ yielded the highest mean QoE in their respective validation comparisons. These values were therefore fixed before the final 30-seed evaluation. The analysis is intended to assess parameter robustness and select a reasonable operating configuration rather than to perform an exhaustive hyperparameter search.

Table 2. Parameter sensitivity on validation seeds 200 - 207.

Validation setting	Mean QoE	Mean latency (s)
$\gamma = 0.2$	0.5675	0.4971
$\gamma = 0.5$	0.5706	0.4944
$\gamma = 0.8$	0.5663	0.4993
$\gamma = 1.1$	0.5731	0.4917
$\gamma = 1.4$	0.5716	0.4941
$K = 4$	0.5728	0.4938
$K = 8$	0.5744	0.4940
$K = 12$	0.5716	0.4941
$K = 16$	0.5747	0.4911
$\eta = 0$	0.5784	0.4921
$\eta = 0.01$	0.5781	0.4907
$\eta = 0.02$	0.5716	0.4941
$\eta = 0.05$	0.5725	0.4930

5.3. Baselines, Matched-Budget Protocol, and Statistical Analysis

MFO is used as the direct ablation baseline because it corresponds to the population-search backbone without the selective refinement stage. HGSTO is implemented as a source-guided adaptation of the genetic-algorithm and simulated-annealing structure reported in [24] and mapped to the same ternary Device-

MEC–Cloud decision space. The implementation uses tournament selection with tournament size 3, uniform crossover with a component-wise probability of 0.5, 10% elitism, an initial mutation probability of 0.30 that decreases linearly over generations, four hill-climbing trials when mutation is activated, and simulated-annealing refinement with initial temperature $T_0 = 0.05$, cooling factor 0.90, and five neighborhood steps per generation. Discrete PSO uses integer-valued positions, inertia weight $\omega = 0.729$, cognitive and social coefficients $c_1 = c_2 = 1.49445$, and velocities rounded and clipped to the interval $[-2, 2]$ before position updates. Greedy EDF (Earliest-Deadline-First) serves as a deterministic low-cost reference: tasks are processed in increasing deadline order, and each task is assigned to the execution layer that yields the best current global score among the three alternatives.

The main stochastic comparison uses a matched target budget of 11,250 fitness evaluations (NFE). The proposed method, MFO, HGSTO, and discrete PSO each use exactly 11,250 evaluations. Greedy EDF requires only 1200 evaluations and is therefore treated as a low-cost deterministic reference rather than as an equal-budget population-based competitor.

All stochastic methods are evaluated on the same 30 workload seeds (0 - 29). For workload seed s , PiFBOA and MFO use solver seed $900000 + s$; PiFBOA additionally uses $1900000 + s$ for the refinement stage, while HGSTO and discrete PSO use $3000000 + s$ and $4000000 + s$, respectively. Thus, the same workload instance is reused across methods, while solver randomizations remain independent. Paired t-tests and Wilcoxon signed-rank tests are used for pairwise comparisons. Holm correction is applied across the four proposed-method-versus-baseline comparisons for each reported metric. In addition, 95% confidence intervals for paired differences and Cohen's d_z are reported. Convergence is plotted against the number of fitness evaluations rather than iteration count so that search progress is compared on a common computational-effort axis.

6. Results and Discussion

6.1. Matched-Fitness-Evaluation Comparison

Table 3 summarizes the main comparison over the 30 paired 400-user workloads. Under the matched search budget of 11,250 fitness evaluations, PiFBOA achieves a mean QoE of 0.5769, compared with 0.5719 for MFO, 0.5242 for HGSTO, and

Table 3. Main comparison over 30 paired 400-user workloads.

Method	QoE	Latency (s)	Energy (J)	Score	NFE
PiFBOA	0.5769 ± 0.0191	0.4926 ± 0.0144	0.0933 ± 0.0036	0.5769 ± 0.0191	11,250
MFO	0.5719 ± 0.0205	0.4927 ± 0.0181	0.0936 ± 0.0041	0.5719 ± 0.0205	11,250
HGSTO	0.5242 ± 0.0218	0.9757 ± 0.0761	0.1253 ± 0.0076	0.5242 ± 0.0218	11,250
Discrete PSO	0.3643 ± 0.0173	1.2553 ± 0.1003	0.1503 ± 0.0085	0.3643 ± 0.0173	11,250
Greedy EDF	0.5383 ± 0.0214	0.5111 ± 0.0182	0.0968 ± 0.0041	0.5383 ± 0.0214	1200

0.3643 for discrete PSO. Greedy EDF achieves a mean QoE of 0.5383 using only 1200 evaluations. PiFBOA and MFO exhibit nearly identical mean latency, at 0.4926 s and 0.4927 s respectively, while their mean device-side energy consumptions are also very close, at 0.0933 J and 0.0936 J.

Figure 1 compares convergence using the number of fitness evaluations as a common search-effort axis. PiFBOA maintains a small but persistent advantage over MFO after the early stage of the search. HGSTO improves more gradually, whereas discrete PSO converges to a lower QoE. Using NFE rather than iteration count provides a more comparable view of search progress across methods with different internal update structures.

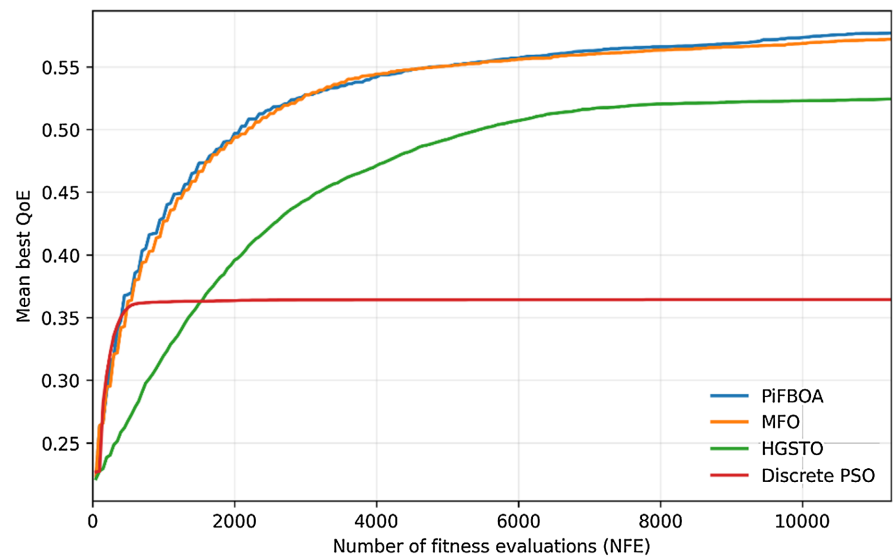


Figure 1. Mean best QoE versus number of fitness evaluations.

6.2. Statistical Robustness

Table 4 reports the paired QoE comparisons between PiFBOA and each comparator. The mean QoE difference between PiFBOA and MFO is 0.0050, with a 95% confidence interval of [0.0023, 0.0077]. The difference is statistically significant under both the paired t-test ($p = 7.31 \times 10^{-4}$) and the Wilcoxon signed-rank test ($p = 7.29 \times 10^{-4}$), and remains significant after Holm correction. In contrast, the differences in mean latency and device-side energy are not statistically significant, with Holm-adjusted p-values of 0.960 and 0.352, respectively.

Table 4. Paired QoE comparisons: PiFBOA minus comparator.

Comparator	ΔQoE	95% CI	t-test p	Wilcoxon p	Holm-adjusted p	Cohen's d_z
MFO	0.0050	[0.0023, 0.0077]	7.31×10^{-4}	7.29×10^{-4}	7.31×10^{-4}	0.69
HGSTO	0.0528	[0.0490, 0.0565]	5.72×10^{-23}	1.70×10^{-6}	1.14×10^{-22}	5.29
Discrete PSO	0.2126	[0.2062, 0.2190]	1.40×10^{-33}	1.73×10^{-6}	5.61×10^{-33}	12.46
Greedy EDF	0.0386	[0.0363, 0.0409]	5.53×10^{-25}	1.69×10^{-6}	1.66×10^{-24}	6.24

Table 5. Paired latency and energy comparisons: PiFBOA minus comparator.

Metric	Comparator	Difference	95% CI	Holm-adjusted p	Cohen's d_z
Latency	MFO	-0.00009	[-0.00380, 0.00362]	0.960	-0.009
Latency	HGSTO	-0.48305	[-0.51134, -0.45476]	9.01×10^{-25}	-6.38
Latency	Discrete PSO	-0.76268	[-0.79945, -0.72590]	4.77×10^{-27}	-7.74
Latency	Greedy EDF	-0.01848	[-0.02180, -0.01516]	6.30×10^{-12}	-2.08
Energy	MFO	-0.00026	[-0.00082, 0.00030]	0.352	-0.173
Energy	HGSTO	-0.03200	[-0.03449, -0.02950]	2.85×10^{-21}	-4.79
Energy	Discrete PSO	-0.05698	[-0.06047, -0.05350]	4.18×10^{-24}	-6.10
Energy	Greedy EDF	-0.00352	[-0.00401, -0.00303]	1.18×10^{-14}	-2.68

As shown in **Table 5**, the latency and device-side energy differences between PiFBOA and MFO are not statistically significant after Holm correction. In contrast, PiFBOA shows statistically significant differences relative to HGSTO, discrete PSO, and Greedy EDF for both metrics.

6.3. Small-Scale Exact-Optimum Validation

To assess solution quality independently of the selected baselines, exhaustive search was performed on ten separate 9-task instances, for which exact enumeration of the ternary search space remains computationally feasible.

As summarized in **Table 6**, PiFBOA and MFO reached the exact optimum in all ten 9-task instances. Greedy EDF matched the exact optimum in seven of the ten instances. These results show that both stochastic optimizers consistently recover the exact QoE optimum on the tractable small-scale test cases, whereas the greedy heuristic does not always reach the optimum.

Table 6. Small-scale exhaustive validation (9 tasks, 10 seeds).

Method	QoE	Optimization score	Exact-score matches
Exact oracle	0.9889 ± 0.0351	0.9889 ± 0.0351	10/10
PiFBOA	0.9889 ± 0.0351	0.9889 ± 0.0351	10/10
MFO	0.9889 ± 0.0351	0.9889 ± 0.0351	10/10
Greedy EDF	0.9333 ± 0.1405	0.9333 ± 0.1405	7/10

This experiment does not establish global optimality for the 400-user problem. Nevertheless, it shows that the implementation can recover the exact optimum on tractable instances and provides an additional sanity check on the search procedure.

6.4. Interpretation and Limitations

The results indicate that the selective refinement mechanism provides a measurable improvement over the MFO backbone under the considered resource-cou-

pled setting. Because offloaded tasks share uplink bandwidth and MEC processing capacity, individual decisions affect the resources available to other tasks. Within this coupled environment, the refinement stage improves QoE relative to MFO, while the differences in mean latency and device-side energy are not statistically significant.

The improvement over MFO remains modest, which is consistent with the design of the proposed method: MFO is intentionally retained as the main population-search mechanism, while the additional refinement stage acts only on the incumbent best solution. The empirical contribution should therefore be interpreted as evidence that this selective hybridization can improve search quality under the considered offloading model, rather than as evidence of a fundamentally new search paradigm.

The study also has several limitations. Workload parameters are synthetically generated and uniformly sampled rather than derived from measured application traces. Validation using measured IoT application traces and measured 5G channel conditions is left for future work. Uplink bandwidth and MEC computing capacity are shared equally instead of being jointly optimized. Future work will jointly optimize the discrete offloading decisions together with continuous communication and computing-resource allocation. Result-download time is neglected, cloud backhaul delay is fixed, and the energy model covers only the IoT device rather than MEC or cloud infrastructure. HGSTO is evaluated through a source-guided adaptation rather than the authors' original implementation. Finally, exhaustive enumeration is practical only for small task counts, so optimality gaps at the 400-user scale remain unknown.

7. Conclusions

This study investigated a selective hybrid metaheuristic for resource-coupled Device-MEC-Cloud task offloading in 5G-enabled IoT systems. The proposed approach retains MFO as the main population-search mechanism and introduces a PFOA-derived refinement stage around the incumbent best solution. The refinement is accepted only when it improves the resource-coupled objective after ternary decoding.

Using a matched search budget of 11,250 fitness evaluations across 30 paired 400-user workloads, the proposed method achieved a QoE satisfaction ratio of 0.5769, a mean latency of 0.4926 s, and a mean device-side energy consumption of 0.0933 J. QoE remained significantly higher than that of MFO under both Wilcoxon and Holm-corrected comparisons, while the differences in mean latency and device-side energy were not statistically significant. The method also achieved higher QoE than the adapted HGSTO, discrete PSO, and the low-cost greedy reference. On ten tractable 9-task instances, it recovered the exact exhaustive-search optimum in all cases.

These results indicate that the selective refinement strategy can improve QoE under shared uplink and MEC resource constraints without introducing a signif-

ificant additional device-side energy cost relative to MFO. Future work will consider the joint optimization of discrete offloading and continuous resource allocation, measured workload and radio traces, and larger-scale optimality analysis.

Author Contributions

Conceptualization, K.T.L. and K.K.D.; methodology, K.T.L. and K.K.D.; software, K.T.L. and B.R.D.J.; validation, K.T.L., K.K.D., and B.R.D.J.; writing-original draft preparation, K.T.L.; writing-review and editing, K.T.L., B.R.D.J., K.K.D., and A.O.; supervision, A.O. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

- [1] Umer, A., Ali, M., Daud, A., Hussain, L., Bukhari, A. and Jehangiri, A.I. (2025) Fault Tolerant & Priority Basis Task Offloading and Scheduling Model for IoT Logistics. *Alexandria Engineering Journal*, **110**, 400-419. <https://doi.org/10.1016/j.aej.2024.10.018>
- [2] Cumia, M., Mujica, G. and Portilla, J. (2025) Extending Battery Lifespan in IoT Extreme Sensor Networks through Collaborative Reinforcement Learning-Powered Task Offloading. *Internet of Things*, **31**, Article 101534. <https://doi.org/10.1016/j.iot.2025.101534>
- [3] Abidi, M.H., Alkhalefah, H., Moiduddin, K., Alazab, M., Mohammed, M.K., Ameen, W., *et al.* (2021) Optimal 5G Network Slicing Using Machine Learning and Deep Learning Concepts. *Computer Standards & Interfaces*, **76**, Article 103518. <https://doi.org/10.1016/j.csi.2021.103518>
- [4] Lohiya, R. and Thakkar, A. (2021) Application Domains, Evaluation Data Sets, and Research Challenges of IoT: A Systematic Review. *IEEE Internet of Things Journal*, **8**, 8774-8798. <https://doi.org/10.1109/jiot.2020.3048439>
- [5] Tong, Z., Wang, J., Mei, J., Li, K., Li, W. and Li, K. (2023) Multi-Type Task Offloading for Wireless Internet of Things by Federated Deep Reinforcement Learning. *Future Generation Computer Systems*, **145**, 536-549. <https://doi.org/10.1016/j.future.2023.04.004>
- [6] Escolar, A.M., Alcaraz-Calero, J.M., Salva-Garcia, P., Bernabe, J.B. and Wang, Q. (2021) Adaptive Network Slicing in Multi-Tenant 5G IoT Networks. *IEEE Access*, **9**, 14048-14069. <https://doi.org/10.1109/access.2021.3051940>
- [7] Li, S., Xu, L.D. and Zhao, S. (2018) 5G Internet of Things: A Survey. *Journal of Industrial Information Integration*, **10**, 1-9. <https://doi.org/10.1016/j.jii.2018.01.005>
- [8] Wang, N., Wang, P., Alipour-Fanid, A., Jiao, L. and Zeng, K. (2019) Physical-Layer Security of 5G Wireless Networks for IoT: Challenges and Opportunities. *IEEE Internet of Things Journal*, **6**, 8169-8181. <https://doi.org/10.1109/jiot.2019.2927379>
- [9] Abrol, A. and Jha, R.K. (2016) Power Optimization in 5G Networks: A Step towards Green Communication. *IEEE Access*, **4**, 1355-1374. <https://doi.org/10.1109/access.2016.2549641>
- [10] Al-Turjman, F. and Alturjman, S. (2018) 5G/IoT-Enabled UAVs for Multimedia De-

- livery in Industry-Oriented Applications. *Multimedia Tools and Applications*, **79**, 8627-8648. <https://doi.org/10.1007/s11042-018-6288-7>
- [11] Park, H. and Lim, Y. (2020) Reinforcement Learning for Energy Optimization with 5G Communications in Vehicular Social Networks. *Sensors*, **20**, Article 2361. <https://doi.org/10.3390/s20082361>
 - [12] Humayun, M., Jhanjhi, N., Alruwaili, M., Amalathas, S.S., Balasubramanian, V. and Selvaraj, B. (2020) Privacy Protection and Energy Optimization for 5G-Aided Industrial Internet of Things. *IEEE Access*, **8**, 183665-183677. <https://doi.org/10.1109/access.2020.3028764>
 - [13] Hussein, M.K. and Mousa, M.H. (2020) Efficient Task Offloading for IoT-Based Applications in Fog Computing Using Ant Colony Optimization. *IEEE Access*, **8**, 37191-37201. <https://doi.org/10.1109/access.2020.2975741>
 - [14] Wang, B., Wang, C., Huang, W., Song, Y. and Qin, X. (2020) A Survey and Taxonomy on Task Offloading for Edge-Cloud Computing. *IEEE Access*, **8**, 186080-186101. <https://doi.org/10.1109/access.2020.3029649>
 - [15] Ebrahim, M.A., Ebrahim, G.A., Mohamed, H.K. and Abdellatif, S.O. (2022) A Deep Learning Approach for Task Offloading in Multi-UAV Aided Mobile Edge Computing. *IEEE Access*, **10**, 101716-101731. <https://doi.org/10.1109/access.2022.3208584>
 - [16] Gu, X., Nahrstedt, K., Messer, A., Greenberg, I. and Milojicic, D.S. (2003) Adaptive Offloading Inference for Delivering Applications in Pervasive Computing Environments. *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, Fort Worth, 26 March 2003, 107-114.
 - [17] Li, X., Qin, Y., Zhou, H., Cheng, Y., Zhang, Z. and Ai, Z. (2019) Intelligent Rapid Adaptive Offloading Algorithm for Computational Services in Dynamic Internet of Things System. *Sensors*, **19**, Article 3423. <https://doi.org/10.3390/s19153423>
 - [18] Mahmood Malik, U., Awais Javed, M., AlMohimeed, A., Alkhathami, M. and Almu-jalli, A. (2024) On the Partial Offloading to Multiple Helpers-Based Task Offloading for IoT Networks. *IEEE Access*, **12**, 123843-123855. <https://doi.org/10.1109/access.2024.3452588>
 - [19] Eang, C., Ros, S., Kang, S., Song, I., Tam, P., Math, S., *et al.* (2024) Offloading Decision and Resource Allocation in Mobile Edge Computing for Cost and Latency Efficiencies in Real-Time IoT. *Electronics*, **13**, Article 1218. <https://doi.org/10.3390/electronics13071218>
 - [20] Yang, S., Lee, G. and Huang, L. (2022) Deep Learning-Based Dynamic Computation Task Offloading for Mobile Edge Computing Networks. *Sensors*, **22**, Article 4088. <https://doi.org/10.3390/s22114088>
 - [21] Wang, J., Hu, J., Min, G., Zhan, W., Zomaya, A.Y. and Georgalas, N. (2022) Dependent Task Offloading for Edge Computing Based on Deep Reinforcement Learning. *IEEE Transactions on Computers*, **71**, 2449-2461. <https://doi.org/10.1109/tc.2021.3131040>
 - [22] Hashim Kareem, Z., Qais Malik, R., Jawad, S. and Abedi, F. (2025) Reinforcement Learning-Driven Task Offloading and Resource Allocation in Wireless IoT Networks. *IEEE Access*, **13**, 79314-79330. <https://doi.org/10.1109/access.2025.3563059>
 - [23] Nieto, G., de la Iglesia, I., Lopez-Novoa, U. and Perfecto, C. (2024) Deep Reinforcement Learning Techniques for Dynamic Task Offloading in the 5G Edge-Cloud Continuum. *Journal of Cloud Computing*, **13**, Article No. 94. <https://doi.org/10.1186/s13677-024-00658-0>
 - [24] Ben Sada, A., Khelloufi, A., Naouri, A., Ning, H. and Dhelim, S. (2024) Hybrid Me-

- taheuristics for Selective Inference Task Offloading under Time and Energy Constraints for Real-Time IoT Sensing Systems. *Cluster Computing*, **27**, 12965-12981. <https://doi.org/10.1007/s10586-024-04578-1>
- [25] Chen, L., Zhang, D., Zhang, J., Zhang, T., Wang, W. and Cao, Y. (2023) A Novel Offloading Approach of IoT User Perception Task Based on Quantum Behavior Particle Swarm Optimization. *Future Generation Computer Systems*, **141**, 577-594. <https://doi.org/10.1016/j.future.2022.12.016>
- [26] Cao, S., Qian, Q., Cao, Y., Li, W., Huang, W. and Liang, J. (2023) A Novel Meta-Heuristic Algorithm for Numerical and Engineering Optimization Problems: Piranha Foraging Optimization Algorithm (PFOA). *IEEE Access*, **11**, 92505-92522. <https://doi.org/10.1109/access.2023.3267110>
- [27] Hamadneh, T., Kaabneh, K., AbuFalahah, I., Bektemyssova, G., Shaikemelev, G., Umutkulov, D., *et al.* (2024) Magnificent Frigatebird Optimization: A New Bio-Inspired Metaheuristic Approach for Solving Optimization Problems. *Computers, Materials & Continua*, **80**, 2721-2741. <https://doi.org/10.32604/cmc.2024.054317>
- [28] Keshta, I., Soni, M., Deb, N., Singh, S., Saravanan, K. and Raza Khan, D.I. (2024) Game Theory-Based Optimization for Efficient IoT Task Offloading in 6G Network Base Stations. *Measurement: Sensors*, **33**, Article 101184. <https://doi.org/10.1016/j.measen.2024.101184>