Benchmarking Bio-Inspired Metaheuristics for Load-Aware Task Offloading in Hierarchical IoT-Fog-Cloud Ecosystems

Lamia Oualili[1], Mohamed EL Ghmary[2], Hassan Echoukairi[3]
Computer Science Department-Faculty of Sciences, Mohammed V University in Rabat, Morocco[1, 3]
ENSAM, Mohammed V University in Rabat, Morocco[2]

Abstract*—The multi-objective nature of task scheduling for novel latency-sensitive applications in IoT-Fog-Cloud hierarchies presents persistent challenges, where optimizing one QoS metric often degrades another. Although bio-inspired algorithms provide adaptive solutions, existing comparative studies are constrained by narrow algorithmic scopes, limited evaluation metrics, and a notable absence of statistical validation. To bridge this gap, we introduce a novel unified benchmarking framework that systematically evaluates five prominent metaheuristics — Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Bacterial Foraging Optimization (BFO), Ant Colony Optimization (ACO), and Artificial Bee Colony (ABC) — for load balancing within a three-tier architecture using the iFogSim simulator. We evaluate performance across three QoS dimensions (response time, makespan, and load imbalance degree) under three escalating workload intensities on heterogeneous infrastructure, with all experiments repeated over 10 independent runs. Statistical significance is assessed via the Wilcoxon signed-rank test against GA, selected as a widely established and computationally stable baseline. Our findings reveal no universally optimal algorithm. GA consistently maintains mean response times below 30.5 ms across all workloads, while PSO and BFO remain statistically indistinguishable from GA on makespan, highlighting their interchangeability under specific conditions. Conversely, ABC uniquely excels in distribution equity, reducing load imbalance from 2.64% to 1.32% as task density increases. ACO, however, incurs statistically significant penalties across all metrics, suffering from a pronounced cloud-bias that elevates load imbalance to 11.0% —up to eight times higher than its counterparts. Collectively, these results confirm the inherent trade-offs between latency, efficiency, and fairness. This reference benchmark delivers a reproducible, statistically validated performance baseline, offering system architects a clear empirical foundation for adaptive algorithm selection in heterogeneous Fog-Cloud environments.

Keywords*—Load balancing; IoT-Fog-cloud; metaheuristic algorithms; iFogSim; QoS optimization; wilcoxon signed-rank test; task scheduling; edge computing

I. Introduction

Billions of connected devices across healthcare systems, smart cities, industrial automation, and transportation are now pumping out data faster than ever, and most of it needs to be processed right away. Cloud computing can certainly handle the volume; computational power was never really the issue. The problem is latency: when an application needs a response in milliseconds, the round trip to a distant data center simply takes too long. This gap is exactly what fog computing was built to close. By placing processing nodes between the IoT devices and the cloud, it gives rise to a three-tier IoT-Fog-Cloud architecture that keeps time-sensitive workloads closer to where the data is actually generated. Fog computing essentially solves the capacity-latency trade-off by moving computation closer to where the data originates, which cuts down on bandwidth usage and makes real-time processing at the network edge actually possible. The catch is that fog nodes are nowhere near as powerful as cloud servers, so resource management becomes a real challenge. Tasks coming from a mix of heterogeneous IoT devices need to be spread across fog and cloud tiers in a way that balances several competing QoS goals at once, namely response time, makespan, and load distribution, none of which can be optimized in isolation. And this task assignment problem happens to be NP-hard: as the number of tasks and resources grows, the search space explodes exponentially, so finding the exact optimal solution simply isn't feasible at any meaningful scale.

Given this complexity, it's no surprise that load balancing in IoT-Fog-Cloud environments remains particularly hard to get right, as fog nodes vary widely in capability, IoT workloads fluctuate unpredictably, and edge applications often come with strict QoS constraints that leave little margin for error. Deterministic methods like Round Robin sidestep the complexity by spreading tasks evenly across nodes, but that simplicity comes at a cost: it ignores resource heterogeneity entirely, which inevitably hurts performance. Metaheuristic algorithms inspired by nature — PSO, ACO, GA, BFO, ABC — have emerged as a more adaptive answer, designed to navigate large, complex solution spaces and converge on near-optimal task assignments across the fog-cloud hierarchy. Yet despite a fair amount of research exploring these algorithms individually, very few studies have actually tested them side by side under the same conditions, using the same metrics, with results backed by rigorous statistical analysis. Most existing studies evaluate only a limited number of algorithms at a time, each under its own topology, metrics, and experimental conditions, which makes meaningful cross-study comparison difficult. Furthermore, many comparative works rely on a single QoS metric, overlooking the multi-dimensional trade-offs that characterize real fog deployments. Because statistical validation across multiple independent runs is rarely part of the picture, the reliability of reported results suffers further, leaving practitioners with little solid, evidence-based guidance for

choosing an algorithm in heterogeneous fog environments. This work sets out to address these gaps not by proposing a new load balancing algorithm, but by conducting a comprehensive, rigorous comparison of five existing algorithms, evaluated under a unified experimental framework using the iFogSim simulator.

The main contributions of this study are as follows:

- A unified experimental framework: all five algorithms are tested under the same conditions; the same three-tier topology, same task distribution, same simulation parameters, so the comparisons actually mean something and can be reproduced.
- Multi-metric evaluation: rather than focusing on a single number, three QoS metrics are tracked at once, namely response time, makespan, and load imbalance degree, to capture the trade-offs that really shape fog deployments in practice.
- Statistical rigor: each algorithm is run 10 times independently, with results reported as mean and standard deviation, giving a much more reliable picture of performance than what a single run could ever show.
- BFO in an IoT-Fog-Cloud setting: Bacterial Foraging Optimization is tested here within a three-tier IoT-Fog-Cloud architecture, a pairing that only a handful of existing studies have actually looked into.

The remainder of this study is structured as follows. Section II reviews related work on load balancing and task scheduling in fog and cloud environments. Section III presents the system model and the experimental framework adopted in this study. Section IV reports the experimental results along with the corresponding statistical analysis, and discusses the main findings and their practical implications. Section V concludes the study and outlines directions for future research.

II. Related Work

Load balancing in IoT-Fog-Cloud environments has attracted considerable research attention, driven by the need to handle latency-sensitive workloads across heterogeneous and resource-constrained fog nodes. Researchers have approached the problem from many angles, ranging from simple deterministic strategies to nature-inspired metaheuristic algorithms, each striking a different balance between response time, resource utilization, and scalability. The literature reviewed below is organized into several categories: survey studies, deterministic approaches, PSO-based methods, ACO-based methods, GA-based methods, BFO-based methods, and comparative evaluations, before closing with the research gaps that motivate the present work.

A. Survey Studies on Load Balancing in Fog Computing

Several systematic survey works have contributed to mapping the state of the art in load balancing for fog environments and identifying where the open research questions still lie. Kaur and Aron [1] conducted one of the most comprehensive studies in this space, covering approaches ranging from simple static policies to fairly sophisticated nature-inspired optimization algorithms. Their proposed taxonomy sorts existing methods along several dimensions, including algorithm type, optimization objective, deployment architecture, and evaluation methodology. A key finding of their analysis is that nature-inspired metaheuristics generally outperform deterministic methods once fog deployments become heterogeneous, particularly under dynamic and unpredictable IoT workloads where resource availability and task arrival patterns fluctuate over time. Their work also flags a problem that recurs throughout the literature: the absence of standardized evaluation frameworks, which makes it difficult to trust comparisons drawn across different studies. This observation directly motivates the unified experimental design adopted in the present study, where all five algorithms are evaluated under strictly identical conditions. Aldossary et al. [2] extended this analysis through a more recent systematic literature review specifically focused on load balancing techniques in fog computing, covering architectures, strategies, and emerging trends across a large corpus of studies. Their review systematically categorizes existing approaches and reveals that the vast majority of published works optimize a single QoS metric, most commonly either latency or energy consumption, while neglecting the multi-dimensional trade-offs that characterize real-world fog deployments. Their analysis further highlights that most simulation-based evaluations rely on single runs without statistical validation, which is insufficient to characterize the behavior of stochastic optimization algorithms that can exhibit significant run-to-run variability. These findings strongly support the multi-metric, statistically validated evaluation methodology adopted in the present work.

At a broader architectural scope, Isong and Mamidza [3] surveyed computational task scheduling across the full IoT-Edge-Fog-Cloud continuum, examining scheduling algorithms, their adaptability to dynamic environments, and the research gaps that persist across deployment layers. Their survey points out that while substantial progress has been made on cloud-specific scheduling, fog-specific challenges remain systematically underaddressed, particularly around multi-tier task coordination, node heterogeneity management, and real-time responsiveness under fluctuating workloads. They further note that most existing fog scheduling algorithms are evaluated under simplified topologies that fail to capture the complexity of real-world three-tier IoT-Fog-Cloud deployments, which limits how far the reported results can actually be trusted in practice. From a broader algorithmic perspective, Sinha and Parewa [4] conducted a systematic review of bio-inspired and machine learning-based multi-task scheduling techniques in cloud computing environments, covering a wide range of nature-inspired algorithms including GA, PSO, ACO, BFO, and their hybrid variants. Their analysis confirms that bio-inspired approaches consistently outperform classical heuristics across multiple QoS dimensions and identifies BFO as a relatively underexplored algorithm in scheduling contexts compared to the more widely studied PSO, ACO, and GA families.

B. Deterministic Approaches and Classical Load Balancing

Among deterministic load balancing strategies, Round Robin remains one of the most widely deployed methods in both cloud and fog environments due to its simplicity, predictability, and negligible computational overhead. By distributing incoming tasks uniformly across available resources in a cyclic

fashion, Round Robin requires no prior knowledge of task characteristics or resource capabilities, making it straightforward to implement in any deployment scenario. However, its fundamental limitation lies precisely in this uniformity: fog nodes vary significantly in processing capacity, memory, communication bandwidth, and current load, and assigning equal numbers of tasks to unequal resources inevitably produces severe load imbalance, increased response times, and degraded QoS under heterogeneous workload conditions. This well-documented limitation has driven the transition toward optimization-based scheduling strategies in the fog computing literature. Azmi et al. [5] compared task scheduling approaches in multi-tier fog-cloud computing, from classical methods such as Round Robin and First Come First Served to greedy multi-objective optimization strategies. Testing these under a unified multi-tier topology, they found that classical deterministic methods struggle with workload heterogeneity, producing higher response times and makespan than optimization-based alternatives. They also point out a complication specific to multi-tier setups: decisions made at the fog level ripple downstream and affect cloud-tier resource utilization, which is part of why multi-metric evaluation matters. What their comparison leaves out, though, is any nature-inspired metaheuristic, PSO, ACO, GA, or BFO, so it remains unclear how classical approaches actually stack up against these more sophisticated methods under the same experimental conditions. Singh et al. [6] proposed a fog-cluster based load balancing technique, organizing fog nodes into hierarchical clusters with cluster heads handling local task distribution. The hierarchy pays off in two ways: less inter-node communication overhead and better scalability than a fully centralized approach. Their results back this up, showing better resource utilization and response time than flat Round Robin. What's missing is any comparison against metaheuristic algorithms, and the metrics covered are fairly narrow, so it's hard to say how well their conclusions would hold up in more complex, heterogeneous fog deployments.

C. PSO-Based Approaches for Load Balancing

Particle Swarm Optimization has emerged as one of the most widely investigated metaheuristic algorithms for load balancing and task scheduling in fog and cloud environments. Drawing on the collective behavior of bird flocks and fish schools, PSO models candidate solutions as particles that move through the search space by combining their own best-known position with the swarm's global best. This lets the algorithm explore large, complex solution spaces without needing gradient information, which fits well with the NP-hard task assignment problems typical of fog computing. Goel et al. [7] compared PSO and ACO for workflow scheduling in fog computing using the iFogSim simulator, evaluating both algorithms on makespan and cost within a three-tier IoT-Fog-Cloud topology. Their setup shares our simulation environment and three-layer architecture, but the focus stays on workflow scheduling rather than load balancing itself, leaving a gap our work sets out to fill. The comparison also stops at two algorithms and two metrics, with no statistical validation across multiple runs. Shaik et al. [8] pair PSO with simulated annealing for resource allocation in fog-cloud environments, a sensible match since simulated annealing's probabilistic acceptance of worse solutions gives PSO exactly what it lacks: a way out of local optima, PSO's main weakness. The payoff: more uniform load distribution and faster response times than standalone PSO. Along similar lines, Kumar et al. [9] pair PSO with Bacterial Colony Optimization for fog load balancing instead, reporting makespan reductions of up to 35% over an adaptive inertia-weight PSO variant under heavy workloads. Neither study, however, benchmarks their PSO variant against GA, BFO, ACO, or ABC, so it's unclear how these hybrids would fare against the other algorithms under the same conditions; this is a gap the present comparison addresses directly.

D. ACO-Based Approaches for Load Balancing

Ant colony optimization brings a different metaphor to the table: pheromone trails left by foraging ants, reinforced over time, gradually steering the colony toward better paths. That mechanism translates naturally into task scheduling and load balancing for fog environments, and it is explored extensively. Yakubu and Murali [10] built on this with a resource allocation framework that pairs ACO with priority-based scheduling across IoT-Fog-Cloud tiers, assigning priority levels by deadline sensitivity and computational demand, then letting ACO's pheromone-guided search construct task-to-resource mappings that minimize response time without violating those priorities. Their comparison, however, stays within the ACO family, with no systematic look at other metaheuristics. Load imbalance never shows up in their evaluation at all, even though it's one of the three metrics, alongside response time and makespan, that we track here. Rafique et al. [11] went a step further with NBIHA, a bio-inspired hybrid that folds together multiple swarm-based operators: ACO-style pheromone updates alongside PSO-style velocity-based position updates. NBIHA consistently outperformed standalone ACO and PSO across several QoS metrics, though their evaluation targets a cloud-centric environment without a fog tier and does not provide a systematic comparison across a broader set of bio-inspired algorithms under unified conditions.

E. GA-Based Approaches for Load Balancing

Genetic algorithms form another well-established branch of metaheuristics applied to fog and cloud scheduling, borrowing from natural selection and genetic inheritance to evolve populations of candidate solutions across generations of selection, crossover, and mutation. Saad et al. [12] built a hybrid GA-PSO algorithm for multi-objective task scheduling in fog environments, aimed at big data applications where high volumes of concurrent tasks come with heterogeneous computational demands and tight latency constraints. The hybrid consistently outperformed both standalone GA and standalone PSO across all three QoS metrics in their tests, though ACO and BFO never entered the comparison. Shakir et al. [13] took a different route, presenting a multi-objective genetic algorithm grounded in SPEA2 for load balancing in large-scale IoT-Fog-Cloud health monitoring systems. Their iFogSim simulations show notable drops in latency and network congestion against classical Round Robin and First Fit policies, but the analysis stops at those classical baselines, leaving PSO, ACO, and BFO outside the comparison.

F. BFO-Based Approaches for Load Balancing

Bacterial Foraging Optimization, originally proposed by Passino and inspired by the foraging behavior of Escherichia

coli bacteria, is characterized by four core biological mechanisms: chemotaxis, reproduction, and elimination-dispersal, which collectively provide a distinctive balance between local exploitation and global exploration. Hosseinlou et al. [14] proposed BFO-HH, a hyper-heuristic framework that employs BFO as a high-level controller to dynamically select and combine four low-level scheduling heuristics in cloud data centers. Their experimental evaluation demonstrates consistent and statistically significant outperformance over GA, PSO, ACO, and ABC across all evaluated metrics, with statistical validation over 25 independent runs using paired t-tests. However, their evaluation is conducted in a cloud-only setting without a fog processing layer. Reffad et al. [15] proposed a dynamic adaptive bio-inspired multi-agent system for healthcare task deployment in fog computing environments, combining BFO-inspired adaptive behavioral mechanisms with a distributed multi-agent coordination framework. While their work confirms BFO's adaptability in dynamic fog environments, their evaluation is application-specific and does not include a systematic comparison with other metaheuristic families.

G. ABC-Based Approaches for Cloud and Fog Scheduling

Several recent studies have explored bee-colony-inspired strategies for resource scheduling in fog and cloud environments. Liu et al. [16] proposed a hybrid particle swarm genetic artificial bee colony algorithm (PGABC) combined with PSO-based load balancing (PGABC-PSO) for fog computing resource scheduling, embedding chromosome-crossover operators from the genetic algorithm into the employed-bee and scout-bee phases of the artificial bee colony (ABC) algorithm to address its tendency to fall into local optima. Their two-level architecture used PSO to balance load within a single fog cluster and PGABC to schedule tasks across clusters, reporting delay reductions of up to 28.5% and energy consumption reductions of up to 12.6% compared to standard ABC and PSO. Similarly, Wang [17] introduced a Modified Artificial Bee Colony (MABC) algorithm for dynamic load balancing across virtual machines in cloud computing, integrating genetic crossover and mutation operators to counteract the rapid loss of population diversity that limits the standard ABC algorithm's ability to propagate knowledge of the best solution across generations. Their results showed improvements in cost, energy, and resource utilization relative to existing approaches. While these studies demonstrate the effectiveness of bee-colony and hybrid bee-colony/genetic strategies for fog and cloud resource scheduling, none provides a systematic comparison across multiple bio-inspired metaheuristic families within a unified three-tier IoT-Fog-Cloud framework. This gap is precisely what the present study addresses by evaluating PSO, GA, BFO, ACO, and ABC under identical experimental conditions.

H. Comparative Evaluations and Identified Limitations

Subramoney and Nyirenda [18] carried out a comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments. Their work centers on workflow scheduling rather than load balancing proper, and while it marks a useful step toward systematic benchmarking, the comparison covers only a limited number of algorithms and leaves BFO out entirely. The results also rest on single runs, with no statistical validation across repeated trials. Vispute and Vashisht [19] took a different angle with LETSO, a scheduling algorithm built around energy efficiency for task scheduling in fog computing, benchmarked against the Bee Life Algorithm and Modified PSO. Here too, the scope stays narrow, with a single primary QoS objective and only two competing algorithms in the mix.

I. Research Gap and Motivation

Taken together, the literature reviewed above points to four gaps that this work sets out to close. Comparative studies tend to stop at a handful of algorithms, tested under different topologies, metrics, and simulation setups, which makes any comparison across studies shaky at best. Most published work also looks at just one QoS metric, missing the multi-dimensional trade-offs that actually play out between response time, makespan, and load distribution. BFO, meanwhile, has received far less attention than PSO, ACO, or GA in three-tier IoT-Fog-Cloud settings specifically, with most BFO-based studies confined to cloud-only or application-specific environments. And statistical validation across multiple independent runs is largely absent, which undercuts how much confidence one can place in reported performance numbers. This study takes on all four gaps at once, comparing five load balancing algorithms — PSO, GA, BFO, ACO, and ABC — within a single unified iFogSim framework [20], across three QoS metrics simultaneously, over 10 independent runs, with Wilcoxon signed-rank tests for statistical validation.

III. Experimental Framework

A. System Model and Architecture

The simulation environment models a three-tier IoT-Fog-Cloud architecture consistent with real-world heterogeneous deployments. The architecture comprises three interconnected layers as illustrated in Fig.1.

To evaluate the behavior of the five load balancing algorithms under increasing task load conditions, three workload configurations of increasing density are considered, denoted T1-Small, T2-Normal, and T3-Heavy. Each configuration preserves an identical architectural structure and infrastructure size while varying only the total number of tasks assigned to the system, as summarized in Table I. The tasks-per-device ratio reflects the average number of tasks generated per IoT device in each configuration, computed as the total number of tasks divided by the number of IoT devices, and serves as an indicator of the per-device workload intensity. The total resource count $|R|$ denotes the number of processing resources available for task allocation, namely the 10 fog nodes and 32 cloud cores ($|R| = 42$), and remains constant across all configurations since only the task volume varies, while IoT devices are excluded from $|R|$ as they act as task sources rather than processing resources.

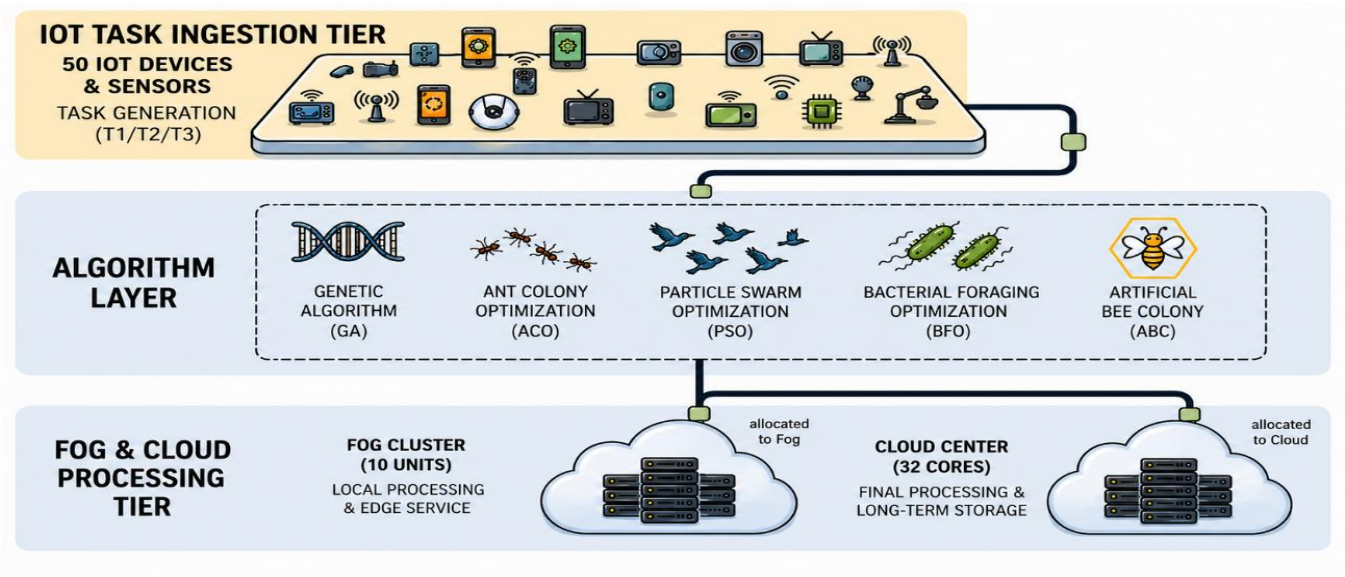

Fig. 1. Three-tier IoT-Fog-Cloud architecture and load balancing.

TABLE I. WORKLOAD CONFIGURATIONS (NUMBER OF DEVICES, NODES, AND TASKS PER SCENARIO)

Parameter	T1-Small	T2-Normal	T3-Heavy
IoT devices	50	50	50
Fog nodes	10	10	10
Cloud cores	32	32	32
Total tasks	500	1000	2000
Type 1 tasks	170	340	680
Type 2 tasks	165	330	660
Type 3 tasks	165	330	660
Tasks/device ratio	10	20	40
Resources |R|	42	42	42

The edge layer comprises fifty heterogeneous IoT devices, configured with varying task generation profiles in terms of rate, size, and computational demand, in order to simulate a realistic and heterogeneous workload. Tasks are generated continuously and deterministically, at a rate that scales with the workload configuration, before being forwarded to the fog layer. Ten heterogeneous fog nodes make up the fog layer, and their configuration does not change across the three workload scenarios. The fog and cloud layers communicate through links defined by latency and bandwidth parameters, mirroring the wide-area network (WAN) conditions one would expect between edge infrastructure and a centralized cloud. Fog nodes were deliberately given varying processing capacities and memory sizes, a choice meant to mirror the kind of resource variability found in real deployments — smart city infrastructure and industrial IoT systems being two common examples.

The cloud layer consists of one cloud server, fixed across all configurations, equipped with 32 high-capacity processing cores that provide substantially greater computational power than the fog nodes. Resource specifications for all three layers are summarized in Table II.

TABLE II. RESOURCE SPECIFICATIONS PER LAYER

Parameter	IoT Devices	Fog Nodes	Cloud Server
MIPS	[50, 100]	[200, 6500]	[50000, 100000]
RAM (MB)	[128, 512]	[512, 8192]	[131072, 262144]
BW (Mbps)	[500, 1000]	[1000, 10000]	100000
Latency (ms)	[5, 10]	[10, 50]	[10, 50]

Tasks are classified into three types with heterogeneous computational requirements, as described in Table III. For illustrative purposes, Type 1 tasks correspond to lightweight operations (e.g., sensing, monitoring), Type 2 tasks to intermediate processing workloads (e.g., data aggregation and filtering), and Type 3 tasks to computationally intensive operations (e.g., video analytics, machine learning inference). The total number of tasks scales from 500 in T1-Small to 1000 in T2-Normal and 2000 in T3-Heavy, maintaining a constant tasks-per-device ratio of 10, 20, and 40, respectively, as detailed in Table I.

TABLE III. TASK TYPE SPECIFICATIONS

Task Type	Length (MI)	Bandwidth (Mbps)	Deadline (ms)
Type 1	[100, 500]	[50, 200]	[10, 50]
Type 2	[501, 1000]	[500, 1000]	[51, 100]
Type 3	[1001, 2000]	[2000, 5000]	[101, 150]

B. Problem Formulation

1) System notation: To formally characterize the load balancing problem, an allocation function $\varphi : T \rightarrow R$ is defined, mapping each task t_i to a resource $R_j \in R$. Three QoS metrics are introduced to quantify the performance of φ in terms of latency, resource utilization balance, and overall completion time, which the allocation function seeks to jointly optimize.

a) Metric 1: Response Time (RT): Response time is the total latency that a task experiences from submission to

completion. It accounts for both network transmission time and execution latency on the assigned resource. For this reason, response time serves as the primary metric for user-facing quality of service (QoS) in latency-sensitive IoT applications [21]. The response time for task t_i assigned to resource R_j is defined as:

$$RT_{\{i,j\}} = lat_j + \frac{len_i.(MIPS_j + BW_j)}{MIPS_j \times BW_j} \quad (1)$$

where, lat_j is the network latency to resource Rj, len_i is the task length in MI, $MIPS_j$ is the processing capacity, and BW_j is the available bandwidth. The average response time over all K tasks is:

$$RT = \frac{1}{K} \sum_{i=1}^{K} RT_{i,\varphi(i)} \times 1000\ (ms) \quad (2)$$

b) Metric 2: Load Imbalance Degree (LID): The Load Imbalance Degree (LID) metric measures how evenly computational load is distributed across the available computing resources. A high LID value indicates that load is unevenly distributed, leading to resource waste and increased queuing delays for tasks assigned to overloaded resources [8]. In this study, LID is computed across all schedulable resources, including both fog nodes and the cloud server, following Shaik et al. [8] [Eq. (18)], as follows:

$$LID = \left(\left(max_{j\in R}(load_j) - min_{j\in R}(load_j)\right) \times 100\ (\%)\right) \quad (3)$$

where, $load_j$ represents the normalized computational load of a resource R_j , defined as the fraction of total work assigned to that resource. A value of LID = 0% indicates perfect load balance.

c) Metric 3: Makespan (MS): Makespan represents the total elapsed time required to complete all tasks in the system, defined as the finishing time of the last task across all resources. It reflects the efficiency of parallel resource utilization — a lower makespan indicates that tasks are well distributed across available resources, minimizing idle time and maximizing throughput [22]. It is defined as:

$$MS = max_{j\in R}\left(\sum_{i:\varphi(i)=j} \frac{len_i}{MIPS_j}\right) \times 1000\ (ms) \quad (4)$$

d) Objective function: The three metrics are combined into a unified scalar fitness function through weighted min-max normalization, allowing simultaneous optimization of all QoS objectives [8]:

$$F(\varphi) = W_{RT}.\widehat{RT} + W_{LID}.\widehat{LID} + W_{MS}.\widehat{MS} \quad (5)$$

where, $\hat{\cdot}$ denotes min-max normalization applied to each metric over the observed value range, ensuring that RT, LID, and MS are scaled to a comparable unit interval before aggregation. The weights are set to W_{RT} =0.40, W_{LID} = 0.30 and W_{MS}=0.30, reflecting the higher priority assigned to response time in latency-sensitive IoT applications, while maintaining balanced consideration of load distribution and completion efficiency.

2) Algorithm configurations and simulation environment: Five bio-inspired load balancing algorithms are evaluated in this study: PSO, GA, BFO, ACO, and ABC. All algorithms are implemented in their standard canonical forms without domain-specific hybridization or parameter tuning, ensuring a fair and unbiased comparison. Parameter values for each algorithm — population/colony/swarm size, iteration count, and algorithm-specific coefficients such as PSO's inertia weight or ACO's evaporation rate — were adopted from their most widely used canonical settings in the metaheuristic optimization literature rather than tuned specifically for this fog-cloud environment; this choice was made deliberately to isolate intrinsic algorithmic behavior from the confounding effect of per-algorithm tuning effort, at the cost of not reflecting the potentially improved performance each algorithm could achieve under a dedicated sensitivity analysis, a trade-off revisited in the Limitations section. Critically, all five algorithms share the same fitness function F(φ), the same task set T, the same resource configuration R, and the same iFogSim simulation environment. Performance differences observed in the results are therefore solely attributable to the intrinsic algorithmic behavior of each method. The complete parameter settings are summarized in Table IV.

TABLE IV. ALGORITHM PARAMETER SETTINGS

Algorithm	Parameter	Value
ABC	Colony size / Iterations	40 / 100
	Abandonment limit (LIMIT)	20
PSO	Particles / Iterations	40 / 100
	Inertia W	0.7
	Coefficients C1, C2	1.5, 1.5
ACO	Ants / Iterations	40 / 100
	α, β	1.0, 2.0
	Evaporation ρ	0.1
	Deposit Q	1.0
GA	Population / Iterations	40 / 100
	Crossover rate	0.8
	Mutation rate	0.1
BFO	Bacteria S	40
	Chemotaxis Nc	100
	Swim length Ns	4
	Reproduction Nre	4
	Elimination Ned	2
	Dispersion prob. Ped	0.25
All algorithms	Fitness function	F(φ) — shared
	Resources |R|	42

All experiments are conducted using iFogSim [20], a Java-based simulator widely used for modeling and evaluating IoT-Fog-Cloud environments. Each algorithm is run 10 times with different random seeds, and results are reported as mean ± standard deviation to account for the stochastic nature of the evaluated algorithms and ensure statistically reliable comparisons.

IV. Results and Discussion

A. T1-Small Topology Results (10 Fog Nodes, 500 Tasks)

Table V presents the comparative performance of the five bio-inspired load balancing algorithms under the T1-Small topology, reporting mean and standard deviation values across 10 independent runs for each of the three QoS metrics.

TABLE V. Performance Results — T1-Small (10 Fog Nodes, 500 Tasks)

Algorithm	Response Time (ms)	Load Imbalance (%)	Makespan (ms)
PSO	30.26 ± 0.95	3.283 ± 0.599	33.76 ± 1.00
GA	30.30 ± 1.25	2.876 ± 0.280	34.27 ± 0.46
BFO	30.14 ± 1.15	2.615 ± 0.376	34.12 ± 0.75
ACO	30.90 ± 1.97	9.701 ± 3.337	39.17 ± 4.61
ABC	30.42 ± 1.52	2.641 ± 0.264	34.80 ± 0.42

1) Response time analysis: Under the T1-Small configuration, response times are consistent across four of the five algorithms, ranging from 30.14 ms for BFO to 30.42 ms for ABC, a difference of only 0.28 ms. BFO achieves the lowest mean response time, marginally outperforming PSO (30.26 ms), GA (30.30 ms), and ABC (30.42 ms), a result attributable to its chemotaxis mechanism, which alternates between local tumble-and-swim exploitation and periodic elimination-dispersal to reliably identify low-latency task-to-resource mappings at this scale. ACO records a mean response time of 30.90 ms, 2.5% higher than BFO, with a standard deviation of 1.97 ms, nearly double BFO's 1.15 ms. In 3 of the 10 runs, ACO's pheromone trails concentrated exclusively on cloud resources, leaving all fog nodes unassigned. This collapse behavior is driven by the cloud's substantially higher MIPS values, which ACO's heuristic interprets as systematically more attractive targets regardless of load distribution.

2) Load imbalance degree analysis: The load imbalance degree results reveal the most pronounced performance divergence among the five algorithms, with ACO standing apart from a tightly clustered group of the remaining four. BFO achieves the lowest mean LID at 2.615%, closely followed by ABC (2.641%) and GA (2.876%), with PSO trailing at 3.283%. These four algorithms differ by less than 0.7 percentage points, indicating broadly comparable load-balancing quality among bacterial foraging, bee colony, genetic, and swarm-based strategies at this scale.

ACO is again the clear outlier, with a mean LID of 9.701%, between 3.3× and 3.7× higher than the other four algorithms, and by far the largest standard deviation observed (3.337%). This instability is directly attributable to the same pheromone-collapse phenomenon identified in the response time analysis: when the colony converges prematurely onto a single resource, load concentrates heavily on that node, while in better-distributed runs the imbalance approaches the level of the other algorithms. ABC's relatively tight standard deviation (0.264%, the lowest of the five) suggests that its food-source/onlooker/scout structure produces more consistent load distribution across independent runs than the other swarm- and evolution-based approaches.

3) Makespan analysis: Makespan results follow a pattern similar to response time, with four algorithms clustered within a narrow range and ACO as a significant outlier. PSO achieves the lowest makespan at 33.76 ms, followed by BFO (34.12 ms), GA (34.27 ms), and ABC (34.80 ms) — a span of just 1.04 ms across the four. Standard deviations for these four range from 0.42 ms for ABC to 1.00 ms for PSO, indicating that ABC produces the most run-to-run stable makespan despite not achieving the lowest mean. ACO records a mean makespan of 39.17 ms with a standard deviation of 4.61 ms, the highest variability observed across all algorithms and metrics in this topology. As with response time and load imbalance, this large spread traces back to the intermittent fog-collapse behavior: runs in which the colony defaults to a single fog node show substantially inflated makespan (up to 44.14 ms), while better-distributed runs remain close to the 33–34 ms range achieved by the other four algorithms.

4) Summary of T1-small findings: Fig. 2 presents a radar chart summarizing the relative performance of the five algorithms across all three QoS metrics at the T1-Small configuration, normalized to [0,1] relative to the best- and worst-performing algorithm on each metric at this scale. The chart shows BFO and ABC forming the largest polygons, reflecting their balanced performance across response time, load imbalance, and makespan even at this Smallest task scale, while ACO already shows a visibly compressed profile driven by its elevated load imbalance (9.701%, roughly 3.7× higher than the next-worst algorithm). PSO and GA occupy an intermediate position, with comparable response time and makespan to BFO and ABC but moderately higher load imbalance. These results establish that no single algorithm dominates uniformly even at the Smallest tested scale, setting the baseline against which the T2-Normal and T3-Heavy configurations are compared.

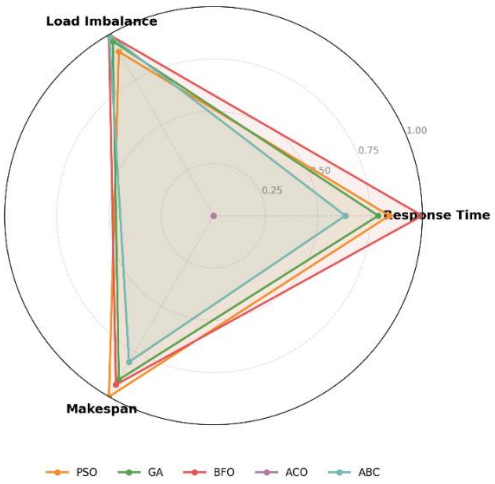

Fig. 2. Normalized radar comparison of bio-inspired load balancing algorithms across QoS metrics — T1-Small (500 tasks).

B. T2-Normal Workload Results (10 Fog Nodes, 1000 Tasks)

Table VI presents the comparative performance of the five bio-inspired load balancing algorithms under the T2-Normal workload configuration, reporting mean and standard deviation values across 10 independent runs for each of the three QoS metrics.

TABLE VI. PERFORMANCE RESULTS — T2-NORMAL (10 FOG NODES, 1000 TASKS)

Algorithm	Response Time (ms)	Load Imbalance (%)	Makespan (ms)
PSO	30.32 ± 1.29	2.758 ± 0.818	34.47 ± 0.57
GA	30.23 ± 1.34	2.234 ± 0.225	34.42 ± 0.47
BFO	30.25 ± 1.14	2.108 ± 0.287	34.15 ± 0.73
ACO	30.04 ± 1.89	9.885 ± 3.429	35.33 ± 1.75
ABC	30.61 ± 1.39	1.956 ± 0.283	34.72 ± 0.28

1) Response time analysis: Under the T2-Normal configuration, response times remain tightly clustered across all five algorithms, ranging from 30.04 ms for ACO to 30.61 ms for ABC, a spread of only 0.57 ms. ACO records the lowest mean response time at this scale, improving slightly over its T1-Small value of 30.90 ms, though this improvement comes with the largest standard deviation among the five algorithms (1.89 ms), reflecting the same convergence instability seen in the T1-Small configuration. GA and BFO remain close to each other at 30.23 ms and 30.25 ms, respectively, both showing marginal gains relative to T1-Small. ABC records the highest mean response time at 30.61 ms, a slight increase from its T1-Small value of 30.42 ms.

2) Load imbalance degree analysis: The load imbalance degree results under T2-Normal preserve the same overall ranking observed at T1-Small, with ABC achieving the best balance among the four closely clustered algorithms and ACO remaining the clear outlier. ABC improves from 2.641% at T1-Small to 1.956% at T2-Normal, the best result among all five algorithms at this scale, and also records the lowest standard deviation (0.283%), reinforcing the consistency advantage noted previously. BFO follows closely at 2.108%, also improving from its T1-Small value of 2.615%, while GA improves modestly from 2.876% to 2.234%. PSO records the highest LID among the four clustered algorithms at 2.758%, alongside the largest standard deviation in that group (0.818%), consistent with its T1-Small behavior of greater run-to-run variability. ACO again stands apart with a mean LID of 9.885%, only marginally higher than its T1-Small value of 9.701%, and continues to exhibit the largest standard deviation of any algorithm-metric combination (3.429%). As observed in the simulation logs, two of the ten runs triggered the " No fog used, defaulting to fog-0" warning, confirming that the pheromone-collapse phenomenon identified at T1-Small persists at higher task densities and continues to drive both the elevated mean and the high variance on this metric.

3) Makespan analysis: Makespan results under T2-Normal show all five algorithms within a relatively narrow band, with BFO achieving the lowest mean at 34.15 ms, followed by GA (34.42 ms), PSO (34.47 ms), and ABC (34.72 ms) — a span of just 0.57 ms across these four. ABC again records the lowest standard deviation (0.28 ms), the most stable makespan of any algorithm at this scale, consistent with its T1-Small behavior. ACO records the highest mean makespan at 35.33 ms, with a standard deviation of 1.75 ms. Notably, this represents a substantial normalization relative to its T1-Small makespan of 39.17 ms (± 4.61 ms): both the mean and the variability decrease considerably at the larger task scale, suggesting that the additional tasks provide more opportunities for the colony to distribute load across fog resources even when individual runs experience partial pheromone collapse, partially offsetting the impact of resource concentration on completion time.

4) Summary of T2-normal findings: The radar chart in Fig. 3 covers the T2-Normal workload, normalizing each algorithm's scores to [0,1] across the three QoS metrics. ABC stands out with a load imbalance of 1.956%, the best figure recorded at this scale, and its polygon is noticeably wider than at T1-Small. ACO manages its lowest response time across all three workloads here, yet its polygon stays narrow because its load imbalance remains at 9.885%. BFO and GA show similar, well-rounded profiles with no particular weakness across the three metrics. PSO's polygon shrinks slightly on the load imbalance side, which matches the higher variability seen in its individual runs. Taken together, the chart shows that the gap separating ACO from the other four algorithms seen at T1-Small has not closed at this intermediate scale.

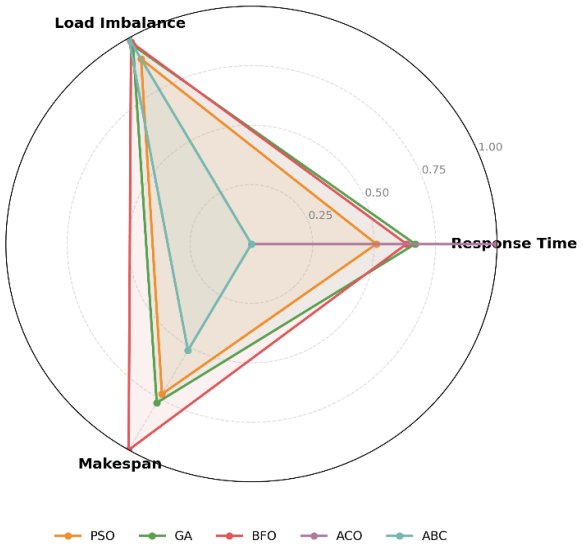

Fig. 3. Normalized radar comparison of bio-inspired load balancing algorithms across QoS metrics — T2-Normal (1000 tasks).

C. T3-Heavy Workload Results (10 Fog Nodes, 2000 Tasks)

Table VII presents the comparative performance of the five bio-inspired load balancing algorithms under the T3-Heavy workload configuration, reporting mean and standard deviation values across 10 independent runs for each of the three QoS metrics.

TABLE VII. PERFORMANCE RESULTS — T3-HEAVY (10 FOG NODES, 2000 TASKS)

Algorithm	Response Time (ms)	Load Imbalance (%)	Makespan (ms)
PSO	30.18 ± 1.10	2.079 ± 1.060	34.38 ± 1.17
GA	30.33 ± 1.21	1.465 ± 0.151	34.51 ± 0.94
BFO	30.40 ± 1.27	1.396 ± 0.144	34.44 ± 0.73
ACO	31.34 ± 2.37	11.019 ± 5.295	38.71 ± 0.75
ABC	30.68 ± 1.28	1.322 ± 0.113	35.12 ± 0.80

1) Response time analysis: Under the T3-Heavy workload configuration, response time results confirm the instability pattern observed for ACO at Smaller scales, now substantially amplified. ACO records the worst mean response time of 31.34 ms, 3.8% higher than PSO (30.18 ms), the best performer at this scale, and its standard deviation of 2.37 ms remains the largest of any algorithm-metric combination across all three topologies. This degradation is consistent with the cloud-bias phenomenon identified at T1-Small and T2-Normal: two of the ten runs at T3-Heavy triggered the " No fog used, defaulting to fog-0" warning, confirming that pheromone trails continue to saturate toward high-MIPS cloud resources at maximum task density, introducing WAN latency overhead that the other four algorithms do not experience.

PSO, GA, BFO, and ABC cluster within a narrow band of 30.18 to 30.68 ms, a spread of only 0.50 ms, indicating that response time converges across fog-dominant strategies regardless of the underlying optimization mechanism once task density reaches 2000. This convergence suggests that the task-to-resource mapping space becomes sufficiently constrained by fixed fog capacity at this scale that algorithm-specific search dynamics exert a diminishing influence on latency outcomes.

2) Load imbalance degree analysis: Load imbalance results under T3-Heavy continue the convergence trend observed across workload configurations for the four non-ACO algorithms, with ABC achieving the best result at this scale. ABC records a mean LID of 1.322%, improving from 1.956% at T2-Normal and 2.641% at T1-Small, and also recording the lowest standard deviation of any algorithm at this scale (0.113%), extending the stability advantage observed at the two Smaller configurations. BFO follows at 1.396%, also improving steadily across configurations (2.615% → 2.108% → 1.396%), while GA records 1.465%, similarly improving from 2.876% and 2.234% at the Smaller scales. PSO records the highest LID among the four clustered algorithms at 2.079%, with by far the largest standard deviation in that group (1.060%) — individual runs ranged from a low of approximately 1.21% to a high exceeding 4.1%, indicating that PSO's particle convergence remains highly sensitive to the random seed even at maximum task density.

ACO records a mean LID of 11.019%, its highest value across all three configurations (9.701% at T1-Small, 9.885% at T2-Normal), with a standard deviation of 5.295%, the largest observed for any metric in this study. Two runs recorded LID values above 16%, directly attributable to the fog-collapse phenomenon, while the better-converged runs remained closer to 6–9%, illustrating the wide behavioral range produced by the heuristic's cloud bias at this task density.

3) Makespan analysis: Makespan results under T3-Heavy show PSO, GA, BFO, and ABC within a comparatively narrow band, with BFO and PSO recording the lowest means at 34.44 ms and 34.38 ms, respectively, followed by GA at 34.51 ms and ABC at 35.12 ms. PSO's standard deviation of 1.17 ms is the highest among these four, while ABC's 0.80 ms remains competitive despite its mean being modestly higher than the other three — consistent with the stability-over-optimality pattern observed for ABC across the load imbalance metric at this and the previous two configurations. ACO again stands as a clear outlier, recording a mean makespan of 38.71 ms with a standard deviation of 0.75 ms. This is 12.0% higher than the next-worst algorithm (ABC, 35.12 ms) and 3.8 percentage points higher in relative terms than ACO's own T2-Normal makespan of 35.33 ms, confirming that the cloud-bias phenomenon's impact on completion time intensifies rather than stabilizes as task density reaches its maximum tested level. Unlike at T1-Small, where ACO's makespan variability was the largest observed (± 4.61 ms), the standard deviation at T3-Heavy is comparatively low, suggesting that at this task density the cloud-routing behavior has become a consistent rather than intermittent feature of the colony's convergence pattern.

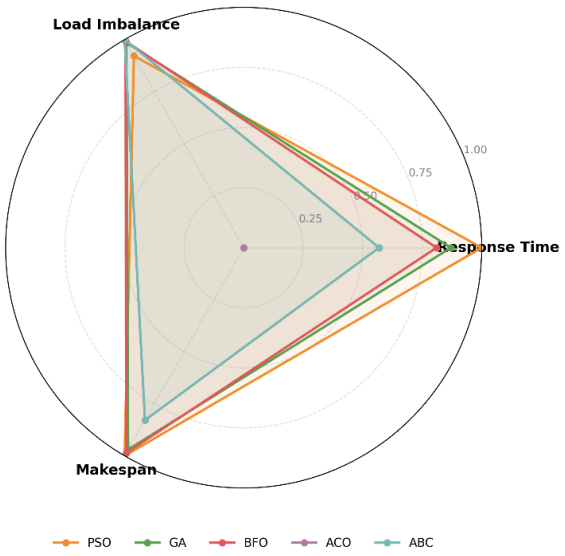

Fig. 4. Normalized radar comparison of bio-inspired load balancing algorithms across QoS metrics — T3-Heavy (2000 tasks).

4) Summary of T3-heavy findings: Fig. 4 presents a radar chart summarizing the relative performance of the five algorithms across all three QoS metrics at the T3-Heavy configuration, normalized to [0,1] relative to the best- and worst-performing algorithm on each metric at this scale. The chart visually confirms ACO's compressed polygon area, driven primarily by its substantially elevated load imbalance and makespan relative to the other four algorithms, while BFO, ABC, GA, and PSO form comparably sized polygons reflecting

their convergence on response time and the progressively tightening gap on load imbalance. Taken together with the T1-Small and T2-Normal results, this final configuration confirms a consistent and widening performance gap between ACO and the other four bio-inspired algorithms as task density increases, while BFO and ABC emerge as the most consistently well-rounded performers across all three tested scales.

D. Cross-Configuration Comparative Analysis

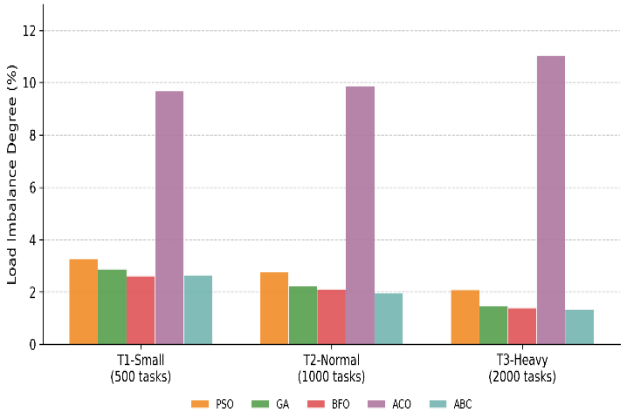

Fig. 5. Load across workload configurations.

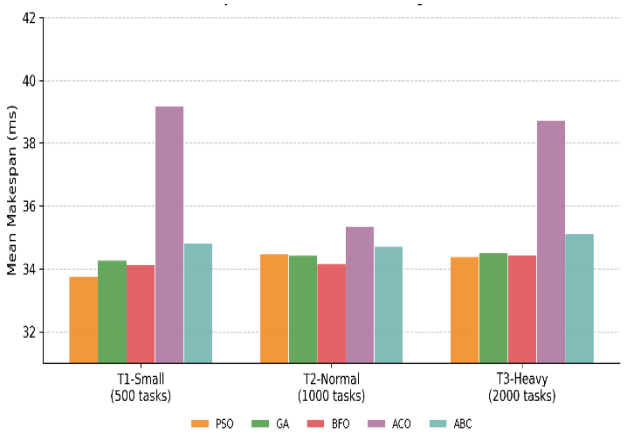

Fig. 6. Makespan across workload configurations.

Fig. 5, 6, and 7 summarize the evolution of the three QoS metrics across the three workload configurations for all five bio-inspired algorithms. Several consistent trends emerge. ABC demonstrates the most stable scalability profile among the five algorithms, recording progressively improving and increasingly stable load imbalance as task density increases (2.641% → 1.956% → 1.322%), while maintaining a moderate and consistent makespan throughout. BFO closely tracks this pattern, achieving comparable or slightly better load imbalance at each configuration alongside competitive response time and makespan, making it the strongest overall performer on response time and makespan jointly. GA shows steady, well-balanced performance across all three configurations without standing out as either a leader or an outlier on any single metric. ACO exhibits consistently degrading and highly variable behavior driven by its cloud-bias phenomenon, with both its load imbalance and makespan worsening as task density increases, and its standard deviations remaining the largest observed for any algorithm throughout the study. PSO shows comparatively higher variability in load imbalance at every configuration, indicating greater sensitivity to random seed and task density than the other four algorithms. Overall, no single algorithm dominates across all metrics and configurations, confirming the inherent multi-objective trade-off nature of the load balancing problem in heterogeneous IoT-Fog-Cloud environments, though ABC and BFO emerge as the most consistently competitive choices across the tested scales. This conclusion holds under the fixed, static fog infrastructure and deterministic task inter-arrival pattern evaluated in this study, and should be revisited under the dynamic, mobile, and larger-scale deployment conditions discussed in the Limitations section.

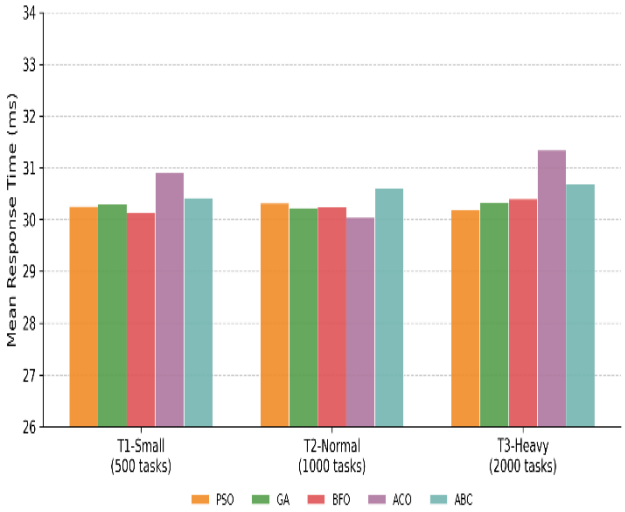

Fig. 7. Response time across workload configurations.

E. Statistical Significance Analysis

To validate the statistical significance of the observed performance differences, a Wilcoxon signed-rank test [23] was applied to compare each algorithm (PSO, BFO, ACO, ABC) against GA across all metrics and workload configurations, using a significance threshold of $\alpha = 0.05$. GA was selected as the reference algorithm for this comparison due to its consistently median performance profile across all nine metric-configuration combinations, neither dominating nor underperforming on any single metric, making it a representative midpoint against which the relative behavior of the remaining four algorithms can be assessed. This non-parametric test was selected due to the Small sample size of 10 independent runs per configuration, which does not justify the normality assumption required by parametric alternatives such as the t-test.

The results reveal several statistically significant patterns. Regarding response time, BFO shows a significant divergence from GA only at T1-Small ($p = 0.002$, lower; Fig. 8), with no significant difference at T2-Normal or T3-Heavy, indicating that its early latency advantage fades as task density increases. ACO and ABC display the opposite pattern, becoming significantly slower than GA at higher task density: ACO at T3-Heavy only ($p = 0.010$, higher; Fig. 10), and ABC at both T2-Normal and T3-Heavy ($p = 0.002$ in both cases, higher; Fig. 9 and 10). PSO

is statistically indistinguishable from GA at T1-Small (p = 0.432; Fig. 8) but becomes significantly different at T2-Normal and T3-Heavy (p = 0.027 and p = 0.010, respectively; Fig. 9 and 10), although the direction of this difference is inconsistent (higher at T2-Normal, lower at T3-Heavy), suggesting the observed differences may not reflect a systematic algorithmic effect.

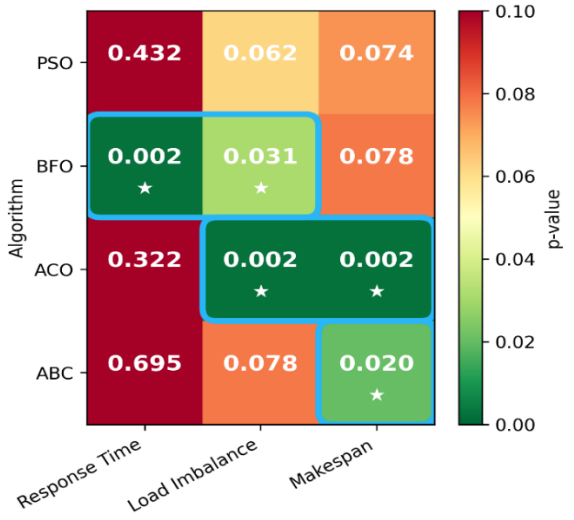

Fig. 8. Heatmap of Wilcoxon signed-rank p-values vs GA for T1-Small workload (500 tasks). Blue borders and ★ denote statistically significant differences ($p < 0.05$).

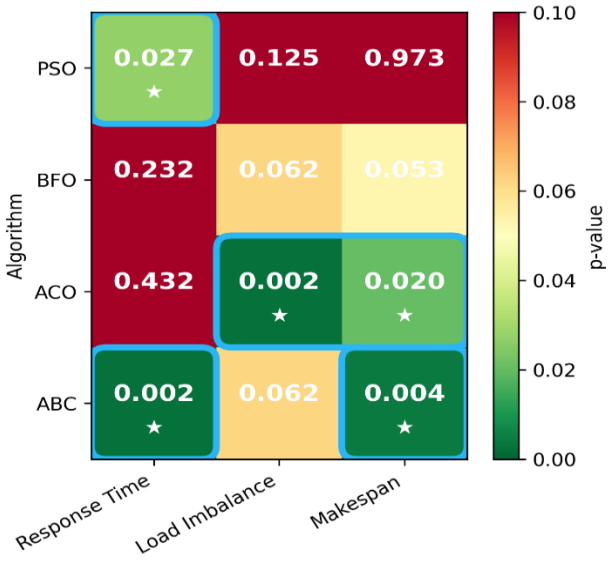

Fig. 9. Heatmap of Wilcoxon signed-rank p-values vs GA for T2-Normal workload (1000 tasks). Blue borders and ★ denote statistically significant differences ($p < 0.05$).

Regarding load imbalance, the pattern is clearly visible across Fig. 8-10. ACO produces significantly higher load imbalance than GA in all three configurations (p = 0.002 in each case), confirming the cloud-bias behavior identified in the qualitative analysis. BFO differs significantly from GA only at T1-Small (p = 0.031, lower; Fig. 8), indicating a brief balance advantage at low task density that does not persist at higher loads (Fig. 9-10). PSO shows no significant load imbalance difference from GA at any configuration (Fig. 8-10), while ABC is statistically indistinguishable from GA at T1-Small and T2-Normal but achieves significantly lower load imbalance at T3-Heavy (p = 0.016; Fig. 10).

Regarding makespan, statistical significance is more selective, as shown in Fig. 8-10. ACO produces a significantly higher makespan than GA across all three configurations ($p \leq 0.020$), and ABC follows the same pattern ($p \leq 0.023$), indicating that GA achieves a measurably lower completion time than both algorithms regardless of task density. PSO and BFO remain statistically equivalent to GA on makespan at every configuration (Fig. 8-10), confirming that GA's makespan performance is indistinguishable from these two algorithms throughout the tested range.

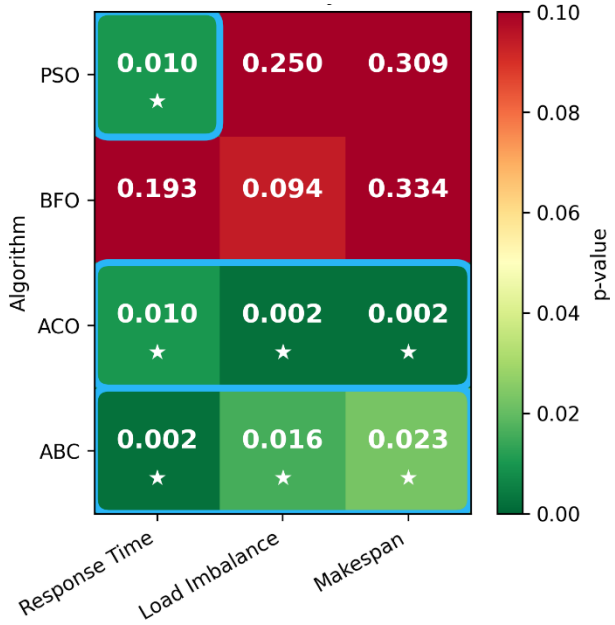

Fig. 10. Heatmap of Wilcoxon signed-rank p-values vs GA for T3-Heavy workload (2000 tasks). Blue borders and ★ denote statistically significant differences ($p < 0.05$).

Overall, these statistical results, summarized across Fig. 8-10, confirm that GA represents a robust and balanced reference point: PSO and BFO remain statistically comparable to GA on makespan throughout, while ACO shows a statistically validated cost relative to GA on all three fronts -significantly higher load imbalance across all configurations, higher response time at high task density (Fig. 10), and higher makespan across all configurations - and ABC trades a modest, significant makespan penalty for load-imbalance performance that is statistically indistinguishable from GA except at the highest task density, where it significantly improves on GA (Fig. 10).

V. Conclusion and Future Work

This study presented a comprehensive comparative evaluation of five bio-inspired load balancing algorithms — PSO, GA, BFO, ACO, and ABC — in a three-tier IoT-Fog-Cloud architecture simulated using iFogSim. The evaluation was conducted across three workload configurations of increasing task density (T1-Small: 500 tasks, T2-Normal: 1000 tasks, T3-Heavy: 2000 tasks) on a fixed infrastructure

comprising 50 IoT devices, 10 fog nodes, and 32 cloud cores, using three QoS metrics: response time, load imbalance degree, and makespan. Statistical significance was assessed through the Wilcoxon signed-rank test against GA as the reference algorithm, across 10 independent runs per configuration.

The results reveal several important findings. First, GA emerges as a robust, balanced reference point across all three configurations, consistently keeping mean response time under 30.5 ms while maintaining moderate, stable load imbalance, with PSO and BFO remaining statistically indistinguishable from GA on makespan throughout the tested range. Second, ABC demonstrates the most stable load distribution among all five algorithms, with load imbalance improving steadily from 2.64% at T1-Small to 1.32% at T3-Heavy, and achieving a statistically significant improvement over GA on this metric at the highest task density, albeit at the cost of a modest but consistent makespan penalty. Third, ACO exhibits a markedly unfavorable profile relative to the other four algorithms, with load imbalance ranging from 9.7% to 11.0% across all configurations, three to eight times higher than its counterparts, alongside a statistically significant cost on response time at high task density and on makespan across all three configurations, driven by a recurring cloud-bias phenomenon in which pheromone-guided search occasionally collapses onto a single resource. Fourth, PSO shows the highest run-to-run variability in load imbalance among the four non-ACO algorithms, suggesting that its velocity-based update mechanism is comparatively more sensitive to task density than the chemotaxis-, evolution-, or bee-colony-based alternatives. Fifth, no single algorithm dominates uniformly across all three QoS dimensions and all three workload scales, confirming the inherent multi-objective trade-off nature of the load balancing problem in heterogeneous IoT-Fog-Cloud environments.

These findings provide actionable guidance for practitioners: GA is recommended as a reliable default choice when response time and makespan are the primary objectives, ABC is preferable when load balance stability is the critical QoS requirement, particularly at higher task densities, and ACO should be approached with caution given its consistent underperformance across all three metrics in this experimental setting, despite the theoretical appeal of pheromone-guided search.

VI. Limitations

Several limitations of this study should be acknowledged.

- The evaluation was conducted on a fixed, static fog infrastructure with deterministic task inter-arrival times and no modeling of node failures, dynamic resource availability, task priorities, or deadline constraints — conditions that may not reflect the variability of real-world IoT deployments; prior work in related wireless sensor network contexts has shown that mobility models can significantly affect the performance of clustering and resource-organization approaches [24], suggesting this is a relevant direction for future investigation.
- Algorithms were evaluated using standard canonical parameter settings without domain-specific tuning or sensitivity analysis, and the computational overhead of the optimization process itself (e.g., per-iteration runtime) was not profiled, which may limit the generalizability of the results to real-time settings.
- The evaluation is limited to response time, load imbalance, and makespan; energy consumption, resource utilization, and throughput were not assessed despite their practical relevance for battery-powered IoT environments.
- Statistical validation used GA as a single reference algorithm rather than exhaustive pairwise comparisons among all five algorithms, which keeps the number of tests interpretable but limits the confirmatory strength of the reported differences.
- The evaluated infrastructure (10 fog nodes, 32 cloud cores) represents a small-to-medium deployment; how algorithm performance and overhead scale at substantially larger infrastructure sizes was not tested.

VII. Future Work

A number of directions are identified for future investigation. Extending the evaluation to heterogeneous and dynamic workload scenarios, with variable task inter-arrival rates, priorities, and deadline constraints, would give a more realistic picture of how each algorithm behaves under practical conditions. Adding energy consumption as a fourth QoS metric would round out the evaluation and bring it closer to the green computing requirements that matter in real IoT-Fog-Cloud deployments. Testing the algorithms under dynamic infrastructure conditions, including fog node failures, resource contention, and device mobility, would shed light on their robustness beyond the controlled simulation setting used here. Applying deep reinforcement learning as an adaptive load balancing strategy and benchmarking it against the five bio-inspired approaches within the same framework is a natural next step toward intelligent and self-adaptive resource management. Finally, the complementary profiles of BFO and ABC suggest that a hybrid BFO-ABC algorithm is worth exploring, with BFO contributing its chemotaxis-based local exploitation and ABC its load balancing stability. A PSO-BFO hybrid, combining PSO's global exploration with BFO's local chemotaxis, represents another promising direction toward better latency and load balancing trade-offs in IoT-Fog-Cloud architectures.

Références

[1] M. Kaur and R. Aron, "A systematic study of load balancing approaches in the fog computing environment," The Journal of Supercomputing, vol. 77, no. 8, pp. 9202–9247, 2021.

[2] D. Aldossary et al., "A systematic literature review on load-balancing techniques in fog computing: Architectures, strategies, and emerging trends," Computers, vol. 14, no. 6, p. 217, 2025.

[3] B. Isong et al., "Computational task scheduling across IoT-edge-fog-cloud continua: Algorithms, adaptability and research gaps," The Indonesian Journal of Computer Science, vol. 15, no. 2, 2026.

[4] R. Sinha, "A systematic review of bio-inspired and machine learning-based multi-task scheduling techniques in cloud computing environments," International Journal of Applied Mathematics, vol. 38, no. 10, pp. 3256–3279, 2025.

[5] Z. Azmi et al., "Comparative analysis of task scheduling in multi-tier fog-cloud computing: From classical approaches to greedy multi-objective

optimization," Journal of King Saud University - Computer and Information Sciences, 2026.

[6] P. Singh et al., "A fog-cluster based load-balancing technique," Sustainability, vol. 14, no. 13, p. 7961, 2022.

[7] G. Goel et al., "Workflow scheduling using optimization algorithm in fog computing," in Proc. Int. Conf. Innovative Computing and Communications (ICICC), vol. 2, 2021, pp. 379–390.

[8] M. Shaik et al., "A hybrid particle swarm optimization and simulated annealing with load balancing mechanism for resource allocation in fog-cloud environments," IEEE Access, vol. 12, pp. 172439–172450, 2024.

[9] A. Kumar et al., "Hybrid bacterial colony optimization and particle swarm optimization for load balancing in fog computing," PLoS One, vol. 21, no. 5, p. e0347176, 2026.

[10] I. Yakubu et al., "An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment," Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 3, pp. 2981–2992, 2023.

[11] H. Rafique et al., "A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing," IEEE Access, vol. 7, pp. 115760–115773, 2019.

[12] M. Saad et al., "Optimizing multi-objective task scheduling in fog computing with GA-PSO algorithm for big data application," Frontiers in Big Data, vol. 7, p. 1358486, 2024.

[13] H. Shakir et al., "A multi-objective genetic algorithm-based load balancing strategy for health monitoring systems in fog-cloud," Computer Systems Science & Engineering, vol. 48, no. 1, 2024.

[14] F. Hosseinlou et al., "Energy-aware priority-based task scheduling in cloud data centers using bacterial foraging optimization," Scientific Reports, 2026.

[15] H. Reffad et al., "A dynamic adaptive bio-inspired multi-agent system for healthcare task deployment," Engineering, Technology & Applied Science Research, vol. 13, no. 1, pp. 10192–10198, 2023.

[16] W. Liu, C. Li, A. Zheng, Z. Zheng, Z. Zhang, and Y. Xiao, "Fog computing resource-scheduling strategy in IoT based on artificial bee colony algorithm," Electronics, vol. 12, no. 7, p. 1511, 2023.

[17] Q. Li and X. Wang, "Modified artificial bee colony algorithm for load balancing in cloud computing environments," International Journal of Advanced Computer Science and Applications, vol. 15, no. 5, p. 1021, 2024.

[18] D. Subramoney et al., "A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments," in Proc. IEEE Symposium Series on Computational Intelligence (SSCI), 2020, pp. 760–767.

[19] S. Vispute et al., "Optimized energy-efficient task scheduling in fog computing," in Int. Conf. Innovations in Computational Intelligence and Computer Vision, Singapore: Springer Nature Singapore, 2022, pp. 735–746.

[20] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, 2017.

[21] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-aware application module management for fog computing environments," ACM Transactions on Internet Technology (TOIT), vol. 19, no. 1, pp. 1–21, 2018.

[22] N. Rizvi et al., "Cost and makespan aware workflow scheduling in IaaS clouds using hybrid spider monkey optimization," Simulation Modelling Practice and Theory, vol. 110, p. 102328, 2021.

[23] F. Wilcoxon, "Individual comparisons by ranking methods," Biometrics Bulletin, vol. 1, no. 6, pp. 80–83, 1945.

[24] H. Echoukairi, A. Kada, and K. Bourgba, "Effect of mobility models on performance of novel centralized clustering approach based on K-means for wireless sensor networks," International Journal of Applied Engineering Research (IJAER), vol. 12, no. 10, pp. 2575–2586, 2017