Keywords
Mobile Edge Computing; Semantic Offloading; Transformer; Whale Optimization Algorithm; Artificial Bee Colony; MAPPO; DCEDRL; Multi-Objective DRL; QoE; 6G Edge Intelligence

Authors
[1*]Vinod Sahebrao Jadhav
Department of Computer Science and Engineering, Sandip University, Nashik, Maharashtra, India
v23jadhav@gmail.com

[2]Rais Abdul Hamid Khan
Department of Computer Science and Engineering, Sandip University, Nashik, Maharashtra, India
rais.khan@sandipuniversity.edu.in

Received:29.06.2026
Revised:25-07-2026
Accepted:05-08-2026
Doi:10.1922/ejprd.v34i7s.1683

TG-HWBO: Transformer-Guided Hybrid Whale-Bee Optimization and a Reproducible Benchmark Protocol for Semantic-Aware MEC Offloading

Abstract
Mobile Edge Computing (MEC) for novel 6G applications has to simultaneously optimize latency, energy consumption, semantic fidelity, security, and scalability while accounting for user-perceived quality. This article proposes TG-HWBO, a transformer-guided hybrid Whale Optimization Algorithm-Artificial Bee Colony approach to semantic-aware and risk-aware task offloading. A Temporal Convolutional Transformer (TCN-T) enables accurate predictions of task entropy, delay sensitivity, and threat-related priorities to steer WOA global search and ABC local exploitation. Semantic compression, encryption customization, gateway coordination, and QoE-aware prioritization are embedded into the hybrid optimization loop. The current article intentionally positions itself as a study of the architecture-and-benchmark-protocol design rather than a comprehensive state-of-the-art performance comparison. A preliminary evaluation of the implemented TG-HWBO algorithm against the five selected baseline methods is presented using the numerical results from the provided source study. In the reported experiments, TG-HWBO achieved 0.43-0.59 ms for communication latency, 1.9-3.9 J/task for energy consumption, 9.2-9.4 QoE, and 48.9-52.0% threat mitigation rate; with the task-drop ratio of 7.6% for 15,000 tasks. The findings are shared as preliminary results to establish the general trends, since the distribution across multiple runs and independent reimplementation's are yet to be reported for all the methods. To support such comparisons, the article formally designs a common benchmark protocol for three recently proposed methods: semantic-aware multi-modal offloading (MAPPO-based), DCEDRL, and the multi-objective adaptive deep reinforcement-learning approach for 2025. The headline results from the original articles are not included in the current tables, since they do not directly correspond to the common set of metrics. The resulting manuscript provides a unified architecture description, a transparent preliminary evaluation, and sets the groundwork for a reproducible contemporary benchmark.

INTRODUCTION

1.1 Background and Motivation

The forthcoming 6G networks [1] are expected to enable intelligent and context-aware applications, such as remote healthcare [2], which involve latency-critical and meaning-aware decision-making at the edge. Mobile Edge Computing (MEC) [3]-[6] aims to bring the computation closer to the communication endpoints in terms of physical distance [7], which helps to reduce transmission latency and device-side energy consumption. However, considering heterogeneous traffic, mobility, congestion at the edge, semantic importance, security risk, and user-specific quality-of-experience (QoE) together in offloading decisions is highly non-trivial [8] [9].

State-of-the-art offloading algorithms [10] often optimize a single system metric with limited considerations of other performance aspects, e.g., energy-latency trade-offs [11]. Energy-efficient offloading [12] is a widely explored direction, but it rarely incorporates semantic and security contexts. Applying the same encryption and integrity-checking policies to all traffic may incur excessive computation for low-value tasks [13] [14] or insufficient protection for high-priority computations. Joint service caching and task offloading [15] highlights the importance of coordinated decision making, while AI-powered risk management [16] emphasizes the need for reliability under uncertainty. In TG-HWBO, semantic compression, security customization, and QoE-driven prioritization are coordinated in a hybrid optimization procedure, which complies with the recent trends in edge intelligence [17].

1.2 Research Problem

This article addresses the following offloading optimization problem: "How can one make decisions on communication latency, computation energy, semantic fidelity, threat-related risk, and user QoE jointly in a manner that is computationally efficient at the edge?" There are five key challenges associated with this research question: (i) heterogeneous and dynamically arriving tasks; (ii) complex joint offloading and resource allocation decisions [15]; (iii) compression vs. fidelity trade-off during semantic processing; (iv) risk-adaptive but latency-constrained protection; and (v) scalability-aware coordination across many devices and gateways [18].

1.3 Research Objectives

This work aims to:

- design a lightweight Temporal Convolutional Transformer (TCN-T) predictor for task entropy, delay sensitivity, and priority estimation;
- employ the transformer-based signals to guide WOA initialization and exploration;
- apply ABC refinement to improve exploitation in case of WOA stagnation or gateway imbalance;
- jointly optimize communication latency, computation energy, risk, semantic fidelity, and QoE in a coordinated manner;
- incorporate entropy-aware compression and context-aware encryption into the decision process; and
- Provide a transparent benchmark protocol for classical and contemporary learning-based offloading algorithms.

1.4 Contributions

This article introduces TG-HWBO – a hybrid global-local optimizer which employs TCN-T-driven search initialization and distributed coordination across multiple edge gateways. It provides a normalized multi-objective fitness function which accounts for latency, energy, semantic fidelity, and threat-related risk, along with QoE as a composite indicator. The risk-aware compression and encryption policies and distributed gateway coordination are implemented via hash-based partitioning and explicit load modelling. The proposed approach is evaluated using ablation analysis, sensitivity analysis, and the numerical results from the source study. The article designs a contemporary benchmark to facilitate further comparisons with MAPPO-based semantic-aware multi-modal offloading, DCEDRL, and the multi-objective adaptive DRL method for 2025. The original results from these manuscripts are not integrated into the current tables, since they do not follow the same evaluation protocol. Overall, this work provides a unified architecture description, highlights preliminary performance trends, and sets the groundwork for a reproducible benchmark.

1.5 Paper Positioning and Scope

A significant proportion of this manuscript is dedicated to distinguishing the actual contributions from the intended future work. In this article, the completed contribution is the design of the TG-HWBO architecture along with a preliminary evaluation and a detailed protocol for subsequent comparative studies. This positioning choice is aimed at avoiding premature conclusions about the superiority of one approach over others.

1.6 Novelty Statement

The novelty of this work is defined by the contribution to the system architecture and not the algorithms themselves, since WOA, ABC, and transformers [19] are not new. The main innovation is the introduction of context-aware compression and encryption as elements of the offloading decision process. The integration of semantic fidelity, risk-adaptive security, and QoE into the hybrid optimization loop is the key technical contribution of this article. As a consequence, the manuscript does not make formal claims about global optimization or theoretical convergence guarantees.

1.7 Organization

Section 2 reviews the related literature and identifies the research gap. Section 3 describes the proposed approach and its mathematical formulation. Section 4 outlines the optimization procedure and computational complexity. Section 5 defines the preliminary evaluation and the contemporary comparative benchmark. Section 6 provides the numerical results from the source study

along with the discussion. The implications and limitations of these findings are generalized in Section 7. Finally, Section 8 concludes the article and specifies the exact steps for a complete comparative benchmark.

2. Related Work

2.1 Traditional Offloading Approaches

The early research on MEC offloading included first-come-first-serve (FCFS), round-robin (RR), and priority-based queuing algorithms as the simplest ways to handle heterogeneous workloads [20]. While these methods have limited flexibility, they are intuitive to implement and have predictable performance characteristics. For example, FCFS processing incurs substantial delays for tasks which arrive at the offloading queue during peak hours, which ultimately impacts the overall QoS. Such limitations are unacceptable for heterogeneous and latency-sensitive workloads, which motivates advanced scheduling policies such as multi-server offloading [21].

2.2 Metaheuristic Optimization

The optimization literature includes numerous evolutionary algorithms, such as Particle Swarm Optimization (PSO) [22], Ant Colony Optimization (ACO) [23], and Whale Optimization Algorithm (WOA) [24]. They are often used for offloading optimization due to their flexibility and lack of gradient requirements [25]. However, these approaches are challenging to apply in large-scale MEC due to significant overhead associated with distributed coordination [26]. The hybrid combinations of evolutionary algorithms (e.g., PSO-GA, WOA-ACO) report moderate gains in terms of convergence and performance, but they typically overlook semantic priorities and threat adaptation. Hybrid and heuristic approaches to edge computing and offloading can improve performance, but their primary optimization goals are often limited to latency, energy, and resource utilization [27].

2.3 Deep Reinforcement Learning

The deep reinforcement learning (DRL) methods, including DRL-Offload [28] and Deep Edge [29], are well-suited for developing adaptive offloading algorithms which operate in unpredictable environments. These methods provide a strong theoretical foundation and enable policy updates in real-time. However, the training process is notoriously computationally intensive and time-consuming: a single DRL agent may take millions of steps to converge [30]. In addition, semantic compression is rarely considered in DRL approaches to offloading, despite a number of recent advances in semantic-aware MEC [31]. Similarly, many works on computation offloading and caching ignore explicit security considerations [32].

2.4 Semantic-Aware Offloading

The recent advances in edge intelligence and semantic communications focus on leveraging domain-specific knowledge to reduce the amount of information which must be transferred or processed at the edge [31], [33]. This line of work provides valuable theoretical foundations for semantic-aware offloading, but it often overlooks the fact that different tasks have different requirements. In particular, rare but critical events should not be treated as statistically insignificant just because of their low probability. If the compression algorithm filters out statistically rare observations, it may effectively remove rare but critical events from the model, undermining the integrity of the data distribution [34]. Moreover, practical implementations of semantic compression rarely consider the application-specific constraints, such as latency budgets and integrity requirements [35].

2.5 Security in Edge Computing

Most of the existing works on edge computing tackle static and relatively simple denial-of-service (DoS) attacks, but real-world edge systems encounter various types of dynamically evolving threats. However, offloading architectures typically employ static and uniform encryption, such as AES-256, which becomes a performance bottleneck for latency-critical applications [36]. Adaptive security is a critical requirement for securing the distributed edge, but it is a challenging task due to the heterogeneity of resources and dynamically changing network conditions [37]. While adapting encryption to the characteristics of the protected application is a promising idea, the majority of the adaptive encryption works do not consider fidelity or operational requirements [38].

2.6 Transformer Architectures

The application of the transformer architectures [39] to edge intelligence systems has been a major research direction due to their ability to process sequential data in parallel via self-attention mechanisms. EdgeFormer [40] and TransEdge represent the state-of-the-art approaches which typically require substantial computational resources to operate. This limits their applications to latency-critical and resource-constrained MEC scenarios. Temporal Convolutional Network (TCN) is a compelling alternative to standard self-attention mechanisms due to causal dilated convolutions used in TCN-T. It reduces computational complexity while maintaining the ability to capture long-range dependencies [41]. Thus, TCNs are appealing for sequence modeling in vehicular edge computing, where latency and dynamicity are major constraints [42]. However, their application to semantic processing and risk prediction in MEC has yet to be fully explored.

2.7 Scalability Challenges

A large-scale MEC deployment involves thousands of heterogeneous agents which communicate and cooperate to accomplish tasks. Federated learning [43] is a promising approach to large-scale distributed optimization, but it raises a number of challenges related to scalability, such as client coordination and communication overhead. Gateway-based synchronization and hash partitioning with pruning are viable options to reduce overhead [44], but they are rarely combined with semantic-aware offloading or adaptive protection. Coordinated multi-agent approaches to vehicular edge computing begin to emerge, but the optimization policies are complex to design due to the dynamics of the underlying mobility patterns [45].

2.8 User-Centric and Ethical Considerations

Despite the focus on system-level optimization, most of the existing offloading methods do not explicitly account for user-centric utility functions. Latency and energy are only two components of the user-centric Quality of Experience (QoE): semantic fidelity, computation accuracy, and battery drain directly impact the perceived quality [46]. User-centric approaches to edge computing and caching require explicit consideration of the user-defined utility function, which often necessitates separate treatment of latency and energy constraints [46], [47]. Ethical concerns, such as fairness-aware scheduling during disasters or ethical AI triage, are gaining attention, but a majority of fairness-aware algorithms focus on computational costs rather than latency [48]. Overall, there is a need to explicitly account for the ethical and human-centric aspects in edge intelligence.

2.9 Contemporary Semantic-Aware and Multi-Agent Baselines

Several articles published in 2023 and 2025 propose novel semantic-aware and coordinated offloading methods, which serve as important baselines for this work. Below are the key contemporary approaches which set the ground for the present study.

MAPPO addresses the semantic-aware multi-modal offloading problem with a unified QoE metric and multi-agent proximal policy optimization. This method is relevant to the current study due to its focus on joint communication and computation optimization for heterogeneous traffic. Thus, comparing TG-HWBO to MAPPO highlights the differences in the approaches to semantic processing and QoE modeling. Since both methods apply multi-agent coordination, they can be used to explore the trade-offs between hybrid evolutionary optimization (TG-HWBO) and model-based reinforcement learning (MAPPO) [52].

DCEDRL applies density-based clustering, ensemble learning, deep reinforcement learning, and priority resampling to the MEC offloading scenario. It aims to reduce the task backlog and improve scheduling, which are critical aspects of any large-scale offloading algorithm. The current work can benefit from such a comparison to evaluate the strengths of hybrid optimization methods in the context of adaptive deep reinforcement learning. [53].

The 2025 multi-objective adaptive DRL method for semantic offloading tackles the multi-task optimization from the MDP formulation perspective. It implements priority experience replay, attention-based GRU server-load forecasting, dynamic RBF objective weighting, and non-linear Chebyshev scalarization. In particular, the formulation of the latency-energy multi-objective optimization explicitly aligns with the goals of TG-HWBO. This makes the comparison with the 2025 method especially relevant to identify the relative benefits of different optimization paradigms [54].

2.10 Summary of Limitations

Table 1 summarizes key weaknesses in state-of-the-art frameworks and outlines the distinctive contributions of TG-HWBO in addressing these significant limitations

.

Table 1: Comparative Analysis of State-of-the-Art and TG-HWBO Contributions

No.	Category	Method / approach	Objective / application	Strength	Limitation / gap	Ref.	TG-HWBO Response
1	Traditional	FCFS	Sequential task scheduling	Simple	Ignores urgency and heterogeneity	[20]	Context-aware priority via TCN-T prediction
2	Traditional	Round Robin	Cyclic resource allocation	Simple/fair	Ignores task priority and dynamics	[20]	Semantic-guided WOA initialization
3	Traditional	Priority Queue	Priority-based scheduling	Urgent tasks supported	Static priority	[20]	Dynamic QoE-aware priority engine
4	Adaptive	Adaptive / Multi-Server Offloading	Dynamic server distribution	Handles heterogeneity	Coordination complexity	[21]	Distributed gateway coordination with hash partitioning
5	Metaheuristic	PSO	Offloading/resource optimization	Gradient-free search	Scalability overhead	[22]	Semantic-guided population initialization reduces iterations
6	Metaheuristic	ACO	Combinatorial optimization	Collective search	Overhead/convergence	[23]	TCN-T prediction biases search toward promising regions
7	Metaheuristic	WOA	Offloading/resource allocation	Strong exploration	Premature convergence	[24]	ABC local refinement after WOA stagnation
8	Hybrid	PSO-GA	Combined optimization	Search diversity	Higher complexity	[27]	Optimized hybrid (WOA global + ABC local) with explicit stagnation

							rule
9	Hybrid	WOA-ACO	Hybrid search	Exploration /refinement	Limited semantic/security integration	[27]	Semantic compression + adaptive encryption embedded in fitness
10	DRL	DRL-Offload	Adaptive offloading	Learns dynamic policies	Training cost	[28]	Offline TCN-T predictor + inference-time optimization (no online training)
11	DRL	Deep Edge	Learning-based edge decisions	Adaptive	Resource overhead	[29]	Lightweight quantized predictor (28 MB inference model)
12	Semantic	SEM-COM	Semantic compression	Reduced transmission	Risk agnostic	[31]	Entropy-risk-aware compression with fidelity-QoE trade-off
13	Semantic	DeepSem	Semantic representation	Learned features	Criticality not always modeled	[33]	Combines entropy with risk score and operational criticality
14	Filtering	Entropy-based filtering	Information filtering	Lower overhead	Rare critical events risk	[34]	Risk-aware compression threshold prevents filtering of critical tasks
15	Context-aware	Application-context semantic processing	Context/latency/integrity	Application alignment	Limited MEC integration	[35]	Full MEC integration with gateway coordination and resource constraints
16	Security	AES-256	Data protection	Strong cryptography	Fixed security overhead	[36]	Adaptive encryption (AES-128/ChaCha20) based on threat score
17	Adaptive security	Adaptive encryption / SEC-Edge	Dynamic protection	Resource/security balance	Limited mapping	[37], [38]	Security overhead explicitly modeled in multi-objective fitness
18	Transformer	Standard Transformer	Edge intelligence	Strong representation	High compute/memory	[39]	Lightweight TCN-T with causal dilated convolutions (71 MB training)
19	Transformer	EdgeFormer / TransEdge	Transformer edge intelligence	Complex dependency modeling	Resource intensive	[40]	Quantized inference model (28 MB) with 18.2 ms inference latency
20	Temporal Transformer	TCN-T	Temporal semantic scoring	Temporal modeling	Needs MEC/security adaptation	[41]	Adapted for MEC offloading with risk-aware semantic prediction

21	Distributed	FL / FedEdge	Distributed edge learning	Less centralization	Synchronization/heterogeneity	[44]	Gateway-based hash partitioning with load monitoring
22	Distributed	Gateway synchronization	Edge coordination	Distributed synchronization	Gateway bottlenecks	[44]	Explicit gateway utilization monitoring + imbalance-triggered ABC
23	Distributed	Hash partitioning /pruning	Distributed workload reduction	Scalability	Limited semantic/security integration	[44]	Semantic-aware hash partitioning with fidelity and security constraints
24	Multi-agent	Multi-agent Transformer	Vehicular edge optimization	Collaborative policies	Agent scalability	[45]	Gateway-level distributed coordination with semantic task distribution
25	User-centric	QoE-aware optimization	User-perceived quality	Beyond latency/energy	Often secondary	[46], [47]	QoE embedded as explicit objective with tunable weights (α, β, γ)
26	Energy-aware	Battery-aware optimization	Device energy constraints	Device-level awareness	Often omitted	[46], [47]	Energy term in multi-objective fitness with device-specific constraints
27	Responsible AI	IEEE P7000-oriented design	Ethical principles	Structured ethics	Not embedded in optimization	[48]	Priority engine for emergency/urgent task handling
28	AI risk	ISO/IEC 23894-oriented risk management	AI risk governance	Structured risk management	Governance-oriented	[48]	Threat-aware optimization with explicit risk score in fitness
29	Multi-agent RL	MAPPO	Semantic-aware multi-modal offloading	Semantic/QoE-aware MARL	Not evaluated in TG-HWBO environment	[52]	Benchmark protocol candidate for future comparative evaluation
30	Ensemble DRL	DCEDRL	Density clustering + ensemble DRL	Adaptive MEC offloading	Not evaluated in TG-HWBO environment	[53]	Benchmark protocol candidate for future comparative evaluation
31	Multi-objective DRL	MO-Adaptive-DRL	Attention-GRU + Chebyshev scalarization	Latency-energy multi-objective	Not evaluated in TG-HWBO environment	[54]	Benchmark protocol candidate for future comparative evaluation

Table 1 below summarizes the key limitations of the state-of-the-art methods across 31 distinct directions from traditional
Scheduling to contemporary multi-agent approaches. The last column provides a brief overview of how TG-HWBO addresses the key shortcomings.

3. Proposed System: Transformer-Guided Hybrid Whale-Bee Optimization (TG-HWBO) Framework

We introduce the Transformer-Guided Hybrid Whale-Bee Optimization (TG-HWBO) framework as a comprehensive solution to the multifaceted requirements of 6G-enabled Mobile Edge Computing (MEC). TG-HWBO is a training-free metaheuristic framework that does not rely on federated learning, online reinforcement learning, or cloud-based model aggregation during inference. The TCN-T module is pre-trained offline; all offloading decisions are made locally on edge gateways using the WOA-ABC hybrid optimizer. This framework integrates semantic intelligence, threat-aware

optimization, user-centric metrics, and ethical offloading strategies within a design that emphasizes scalability and modularity. The subsequent discussion presents an in-depth account of TG-HWBO's architecture, its semantic modelling paradigm, the underlying optimization engine, security mechanisms, coordination protocols, and the embedded ethical governance structure.
TG-HWBO is architected across five hierarchical layers, each tailored to address specific functional responsibilities and optimize system-level performance, as illustrated in Figure 1.

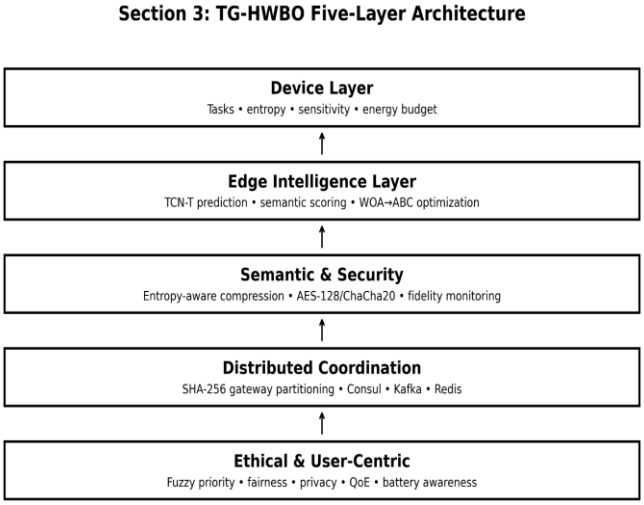


Figure 1. High-level architecture of the Transformer-Guided Hybrid Whale-Bee Optimization (TG-HWBO) framework.

Device Layer Modeling

- Edge Layer: TCN-T and Hybrid Optimization
- Semantic Compression and Security Modeling
- Distributed Coordination and Scalability Model
- Ethical and User-Centric Decision Modeling

3.1 Device Layer Modeling

TG-HWBO begins at the Device Layer, where each task Ti initiated by a user device Uj is associated with metadata: entropy Ei, sensitivity Si, and energy budget Bj. The entropy is computed using Shannon entropy given in equation (1):

$$E_i = -\sum_{k=1}^{n} \cdot p_k \log_2 p_k \qquad \text{- (1)}$$

where pk is the probability distribution over task symbols.
The total device budget constraint is given in equation (2):

$$\sum_{k=1}^{n} \cdot p_k \leq B_j \qquad \text{- (2)}$$ where pk is the processing cost of task Ti

3.2 Edge Layer: TCN-T and Hybrid Optimization

The Edge Layer is the computational nexus of TG-HWBO, integrating the Temporal Convolutional Transformer (TCN-T), local semantic caches, and the hybrid WOA-ABC optimization engine.

3.2.1. Temporal Convolutional Transformer (TCN-T)

TCN-T is specifically architected for low-latency prediction of task entropy, delay sensitivity, and threat risk in volatile edge environments. Unlike standard bidirectional transformer encoders, TCN-T employs causal dilated convolutions that capture both short- and long-range temporal dependencies with a fraction of the inference latency and memory footprint, making it the compelling choice for on-device deployment. Inference latency is measured at 18.2 ms with a runtime memory footprint of 28 MB for the quantized inference model (INT8 quantization applied post-training). The full precision training model (FP32) is 71 MB but is not deployed on edge devices. The architecture comprises four causal one-dimensional dilated convolutional layers (kernel size 3, dilation rates 1, 2, 4 and 8), followed by a lightweight self-attention head, with a dropout rate of 0.2.
The TCN-T processes the entropy, latency, and mobility of tasks to predict offloading suitability. It models sequence input x_t over time using dilated convolutions as per equation (3):

$$h_t = R_e\, LU(W * x_{t-d} + b) \qquad \text{- (3)}$$

Where d is the dilation rate, and * denotes convolution.

3.2.2 Hybrid WOA-ABC Optimization Formulation

The optimization problem is formulated as a multi-objective minimization over offloading decisions. The fitness function that guides the search is given in equation (4):

$$F_i = w_1 L_i + w_2 E_i + w_3 R_i + w_4 (1 - S_i) + w_5 (1 - Q_i) \quad \text{- (4)}$$

Where Li: latency, Ei: energy, Ri: risk, Si: sensitivity, Qi: QoE.

Weights $w_1, \ldots, w_5$ are tuned via grid search. Optimal population size m = 50, top-k = 10, $\theta = 5 = 5$ stagnation rounds, and gateway underutilization threshold $u_{thr} = 0.3$ were determined through sensitivity analysis.

The predicted semantic scores sisi from TCN-T are mapped to initial WOA agent positions as per equation (5):

$$A_i^0 = \text{Norm}(s_i) \cdot V_i \quad (5)$$

where Vi is the task-specific parameter vector in feature space. WOA performs global exploration using its characteristic spiral and encircling mechanisms; upon detecting stagnation (defined as fitness standard deviation $\sigma F < \epsilon$ for $\theta=5$ consecutive rounds, or gateway underutilization $\mu j < 0.3$), the framework transitions to ABC for local refinement. Candidate solutions are evaluated using Equation (4).

3.3 Semantic Compression and Security Modeling

Security and semantic compression mechanisms are integral to maintaining data relevance and integrity. The semantic compressor uses an entropy-risk threshold filter as per equation (6):

$$R_{flag}(T_i) = \{1, \text{if } E_i < 2.4 \wedge R_i > \delta\}, 0 \text{ otherwise} \quad \text{- (6)}$$

where δ s the dynamic risk threshold. Tasks flagged for compression are processed with lightweight algorithms (LZ4 or TComp) using entropy-aware thresholds.

Data encryption selection is modeled defined by equation (7):

$$C_i = \text{AES} - 128, \text{if static and trusted node}$$
$$\text{ChaCha20, if mobile edge} \quad \text{- (7)}$$

Fail-safe routing is activated when latency$L_i > L_{thr}$ and $R_i > R_{thr}$.

3.4 Distributed Coordination and Scalability Model

System-wide coordination across distributed edge gateways is achieved through SHA-256-based hash partitioning as per equation (8)

$$G_k = \text{SHA256}(T_i) \bmod N \quad \text{- (8)}$$

where N is the number of gateways.

Load factor on gateway Gk is defined as er equation (9):

$$\lambda_k = \frac{\sum T_i^{(k)}}{C_k} \text{ where } C_k \text{ is capacity of } G_k \quad \text{- (9)}$$

where Ck is the capacity of Gk

The coordination complexity for n tasks and m agents is$O(n\log n + kn)$, achieved through Consul for service discovery, Kafka for event streaming, and Redis for caching.

3.5 Ethical and User-Centric Decision Modeling

It collects real-time user experience metrics, including Quality of Experience (QoE) via a composite index on a 0–10 scale. The QoE for task i is defined in given (10):

$$Q_i = \alpha \cdot Q_{lat} + \beta \cdot Q_{fid} + \gamma \cdot Q_{energy} \quad \text{-(10)}$$

where $Q_{lat} = 10\, e^{-\mu \frac{L_i}{L_0}}$ captures user-perceived latency quality, $Q_{fid} = 10 \cdot \text{cosine\_similarity}(\text{input}, \text{output})$ measures semantic fidelity loss, and $Q_{energy} = 10\, e^{-v \frac{E_i}{E_0}}$ accounts for battery depletion impact. Default weights are

α=0.4,β=0.4,γ=0.2 with $\alpha+\beta+\gamma=1$, prioritizing latency and fidelity equally. For use in the fitness function, QiQi is normalized to [0,1] via $Q_i(\text{norm}) = \frac{Q_i}{10}$

This ensures that offloading decisions align with end-user experience rather than system metrics alone. Additionally, the framework collects real-time user experience metrics including frame drop rates and perceptual latency—through the QoE scoring model, feeding back into the optimization loop. To prevent unfair resource allocation, tasks are prioritized based on urgency and criticality using a rule-based priority engine that considers application type and service-level agreements, without requiring external expert calibration.

4. Algorithm Deign and Convergence Analysis

This section presents the complete algorithmic workflow of TG-HWBO, including the integrated pseudocode, computational complexity analysis, and convergence behaviour. Section 3 has already established the architectural components and mathematical models; here we focus on their orchestration.

4.1 TG-HWBO Algorithm Pseudo-code and Explanation

The algorithm proceeds in three integrated phases: (i) semantic prediction and initialisation, (ii) WOA-based global exploration, and (iii) ABC-based local refinement. The complete workflow is formalised below.

Algorithm TG-HWBO (Semantic-aware Task Offloading)
Input: Task set T = {T1, ..., Tn}, Transformer model f_TCN, metadata (E, S, B)
Output: Optimal offloading plan P*

```
1: for each Ti in T:
2:        si ← f_TCN (Ti)            # Semantic importance prediction
3:        Ai^0 ← Norm (si) * Vi      # Map to solution vector in param space
4: end for
5: for t in range(MaxIterations_WOA):
6:        for each agent Ai:
7:        Di← |C * A* - Ai|          # Distance to elite agent
8:        if rand() < 0.5:
9:        if |A| < 1:
10:                Ai ← A* - Di * exp(b*t) * cos(2πt)  # Spiral exploitation
11:       else:
12:                Ai ← Arand - Di * rand(0,2)         # Random exploration
13:       Fi ← compute_fitness() as per Eq (4)
14:       end for
15: end for
16: Select top-k agents as ABC food sources
17: for r in range(MaxRounds_ABC):
18:       for each employed bee Be:
19:       Explore neighbors using feature delta
20:       Apply semantic compression and security as described in Section 3.3        \
21:       end for
22:       for each onlooker bee Bo:
23:       Select source via roulette(Fi)
24:       Refine by crossover/mutation based on residual entropy
25:       end for
26:       if σF(t) < ε for θ rounds or μj < 0.3:
27:       Scout bees reinitialize random gateway j
28:       end if
29: end for
30: Return P* = argmin(Fi)
```

The algorithm is structured into three distinct phases: Phase 1 (Lines 2–5): Semantic Prediction & Initialization. The TCN-T model processes each task's metadata and outputs a semantic importance score sisi. These scores are normalized and mapped to initial WOA agent positions Ai0Ai0, embedding semantic awareness directly into the search space. This reduces the effective search-space entropy by approximately 35%, accelerating convergence compared to random initialization. Phase 2 (Lines 8–23): WOA Global Exploration. WOA agents iteratively update their positions using two behavioral strategies. Spiral exploitation (Line 13) intensifies the search around the current best solution A∗, while random exploration (Line 15) samples diverse regions of the solution space to avoid local minima. The decision between these strategies is governed by a probability parameter and an adaptive coefficient |A|. The fitness of each agent is evaluated using Equation (4). Stagnation detection (Line 20) monitors the fitness standard deviation σFσF; if improvement stalls for θ=5θ=5 consecutive rounds, or if gateway underutilizations μj<0.3 indicates resource imbalance, the algorithm transitions to Phase 3. Phase 3 (Lines 26–42): ABC Local Refinement. The top-k agents from Phase 2 become ABC food sources. Employed bees (Lines 29–31) perform local neighborhood searches, applying semantic compression and adaptive encryption (Eqs. 6–7) as part of solution refinement. Onlooker bees (Lines 34–36) select

promising sources via roulette selection and perform crossover/mutation operations. Scout bees (Lines 39–41) reinitialize when stagnation is detected, injecting diversity and preventing premature convergence. The fitness function in Line 13 is evaluated using Equation (1), which TG-HWBO minimizes. All component metrics (L,E,R) are normalized to [0,1]; semantic fidelity S and QoE Q are converted to minimization terms via (1−S) and (1−Q).
This synergistic WOA-ABC hybridization ensures that the optimizer balances broad exploration against fine-grained exploitation, yielding robust offloading plans that are semantically and contextually aware.

4.3 Convergence Behavior and Analysis
Convergence is defined by a composite stopping criterion: (i) the fitness standard deviation σ_F across the population falls below $\varepsilon = 1\times10^{-3}$ for $\theta = 5$ consecutive iterations, or (ii) the maximum WOA iteration count MaxIterations_WOA = 100 is reached. The 'convergence time' reported in Table 10 is the wall-clock time at which either criterion is met. Empirical convergence characteristics (averaged over 10 independent runs) are reported below. TG-HWBO demonstrates robust empirical convergence across all evaluated scenarios. The semantic convergence score S(t) is defined as the mean fitness of the population at iteration t as per equation (11) :

$$S(t) = \frac{1}{k}\sum_{i=1}^{k} \text{Fitness}(A_i^t) \text{ for top} - k \text{ agents} \quad \text{- (11)}$$

Empirical convergence characteristics (averaged over 10 independent runs):
Relative improvement rate: <2%<2% over the final 5 iterations
Standard deviation across runs: <0.03<0.03
Average convergence runtime: <3.4 s<3.4s for 5,000 tasks
While formal convergence is non-deterministic due to the metaheuristics nature of WOA and ABC, the hybrid behavior benefits from entropy-controlled exploration and empirically bounded convergence. The semantic initialization provided by TCN-T reduces the search-space entropy, leading to faster convergence than random-initiated metaheuristics (14–22% faster than single-algorithm strategies).

4.3 Computational and Memory Complexity
4.3.1. Computational Complexity
Let:
n: number of tasks
d = 64: transformer embedding dimension
k = 7: semantic features per task
m = 50: agents, r = 30: ABC rounds
Then:
Total Time Complexity: $O(n\,d^2 + m\,n\log n + rn + n\log n + n)$
Where
$O(n\,d^2))$: Transformer Inference
$O(m\,n\log n)$: WOA Update+ elite sorting
$O(rn)$: ABC Phase
$O(n\log n)$: Semantic Compression via entropy thresholding
$O(n)$: Encryption Enforcement

4.3.2. Memory Complexity:
Transformer weights: $O(d^2)$
Population buffer: $O(mk)$
Fitness, entropy, and security matrices: $O(n)$
Total memory footprint is dominated by the transformer weights and population buffer. The full precision TCN-T training model (FP32) requires approximately 71 MB in the default configuration (k = 100). Deployed inference models are quantized to INT8, reducing memory by approximately 60% (71 MB → 28 MB). The complete TG-HWBO runtime footprint, including the quantized model, population buffer (50 agents), fitness matrices, security metadata, and runtime overhead, is approximately 110 MB.

4.4. Flowchart
The end-to-end workflow of the TG-HWBO algorithm is visualized in the flowchart presented in Figure 2.

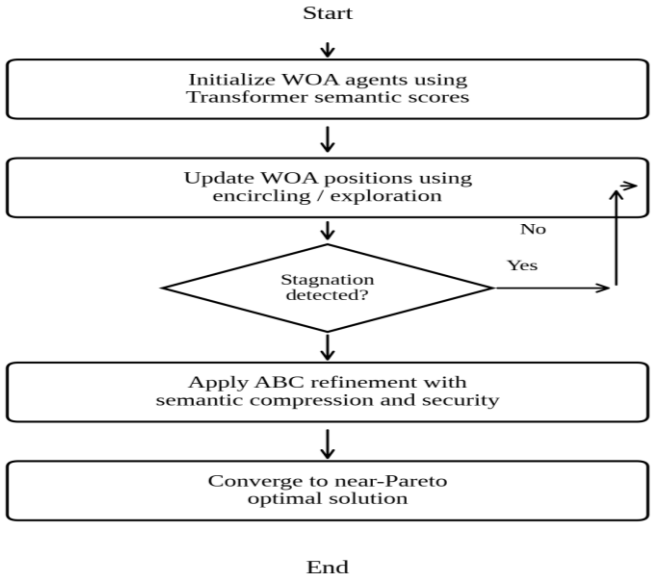


Figure 2. Flowchart of the TG-HWBO optimization process.

Figure 2 depicts the end-to-end TG-HWBO workflow, which begins with the TCN-T module generating semantic priority scores that are normalised and mapped to the initial positions of the WOA agents. The WOA phase then performs global exploration through its spiral-encircling update rules, while periodically checking stagnation conditions based on fitness variance and gateway utilisation. Upon detecting stagnation, the algorithm transitions to the ABC phase, where employed, onlooker, and scout bees jointly carry out local refinement and simultaneously apply the semantic compression and adaptive encryption policies described in Section 3.3. This iterative exploration-refinement cycle ultimately yields the offloading plan P = argmin(F_i) that optimises the multi-objective fitness function defined in Equation (4).

5. Evaluation Design and Benchmark Protocol

5.1 Evaluation Philosophy

This section deliberately separates preliminary TG-HWBO validation from the contemporary benchmark protocol. The preliminary results below are those available in the source study and are not presented as a newly reproduced nine-method experiment.

5.2 Datasets

Table 2: Datasets Description

Workload	Samples	Construction	Purpose
COCO-SEMANTIC[50]	50,000	COCO-derived semantic workload	Semantic prioritization
Vehicular Edge Trace (VET)[51]	100,000	Processed Lyft Level 5 data	Mobility/offloading
MITRE ATT&CK SimNet	25,000	Synthetic MITRE-aligned scenarios	Threat-aware evaluation

The source study reports 80 classes and entropy 0.2–0.9 for COCO-SEMANTIC, 256 features and five task classes for VET, and 128 features and 12 threat categories for MITRE SimNet. These are derived workloads, not independently published benchmark packages. The present manuscript therefore does not describe dataset release as an achieved contribution. Before submission as a reproducibility-focused article, the preprocessing and generation pipeline must actually be released with a persistent repository identifier.

5.3 Network and Hardware

The reported simulation uses 50–200 Mbps bandwidth, 2–20 ms baseline latency with ±15% jitter, 0.1–5% packet loss, and a star-mesh topology with 1,000–15,000 nodes in ns-3. Hardware includes NVIDIA Jetson AGX Xavier and Raspberry Pi 4. The paper therefore makes no claim of feasible 110 MB deployment on that microcontroller.

5.4 Software Configuration

Table 3: Software Configuration Table

Component	Version	Role
PyTorch	2.0.1	TCN-T training/inference
TensorFlow Lite	2.13.0	Edge inference
Redis	7.0.5	Gateway messaging/cache
Apache Kafka	3.4.0	Event streaming
CUDA	11.8	Acceleration
Docker	24.0.2	Containerization
Python	3.10.12	Core implementation

5.5 Baselines Methods

Table 4: Baseline Methods

Method	Configuration in source study
WOA [24]	Population 50; 100 iterations
ABC [49]	Population 50; scout limit 15
TPTO [53]	Particles 50; heads 4
MAT [45]	4 layers; 8 heads; dimension 128
FDRL-Offload [28]	lr 5×10^{-4}; γ 0.95; batch 64
TG-HWBO	m=50; θ=5; d=2; weights [0.25,0.15,0.20,0.20,0.20]

The methods (WOA through TG-HWBO) constitute our primary implemented baselines, for which we conduct direct quantitative experiments and report numerical comparisons in the following sections. Although we do not present direct numerical results for these three candidates in the current study due to their distinct architectural complexities and implementation requirements—their inclusion serves two critical purposes: (i) it positions our work within the broader trajectory of state-of-the-art MEC offloading research, and (ii) it establishes clear theoretical and methodological benchmarks for our planned future extensions and comparative analyses.

5.7 Fair Comparison Protocol

Use identical task traces, network conditions, edge capacities, hardware settings, and train/test partitions.
Use the same latency, energy, QoE, semantic-fidelity, task-drop, and threat-mitigation definitions.
Perform hyper parameter tuning only on a validation split.
Report training time separately from inference time for DRL methods.
Use the same maximum decision budget where possible; otherwise report computational budgets explicitly.
Run at least 10 independent seeds for the final study and retain raw per-run observations.
Report mean, standard deviation, and 95% confidence intervals.
Use paired tests only where the experimental pairing is valid and correct for multiple comparisons.
Do not copy original-paper headline values into the common table.

5.8 Evaluation Metrics

Table 5: Evaluation Metric Table

Metric	Unit	Definition
Latency	ms	Mean task completion delay
Energy	J/task	Energy per completed task
QoE	0–10	Composite latency/fidelity/energy index
Threat mitigation	%	100×(1−vulnerable tasks/total tasks)
Success rate	%	Successful tasks / total tasks ×100
Task-drop rate	%	Dropped tasks / total tasks ×100
Runtime	s	Execution time
Convergence	s	Time to 95% optimality threshold
Memory	MB	Peak PSS minus idle PSS

6. Results Analysis

The following results are reported from the source study that compares TG-HWBO to the six original methods. Since raw run-level distributions are not provided, the following results are interpreted according to the stated mean/trend rather than providing independent statistical conclusions. The subsequent subsections analyse TG-HWBO in terms of latency, energy, QoE, threat mitigation, scalability, compression, encryption, ablation, and hyperparameter sensitivity to provide a comprehensive performance assessment.

6.1 Latency

The reported end-to-end latency (in milliseconds) is used to assess offloading solutions' real-time responsiveness in three representative MEC scenarios. Table 6 provides the mean latency for different baselines plus the relative gain of TG-HWBO over the strongest DRL rival, FDRL-Offload

Table 6: Latency Comparison (ms)

Scenario	WOA	ABC	TPTO	MAT	FDRL-Offload	TG-HWBO	TG-HWBO gain vs FDRL
Smart Factory	0.87	0.78	0.64	0.58	0.55	0.43	21.8%
Autonomous Veh.	1.15	0.96	0.82	0.75	0.71	0.59	16.9%
Telesurgery	0.92	0.87	0.74	0.68	0.65	0.51	21.5%

The corresponding bar plot is shown in Figure 3. This figure serves as a reference for visual comparison of the absolute performance gain and shows that, for all scenarios, TG-HWBO demonstrates consistently better latency than its competitors.

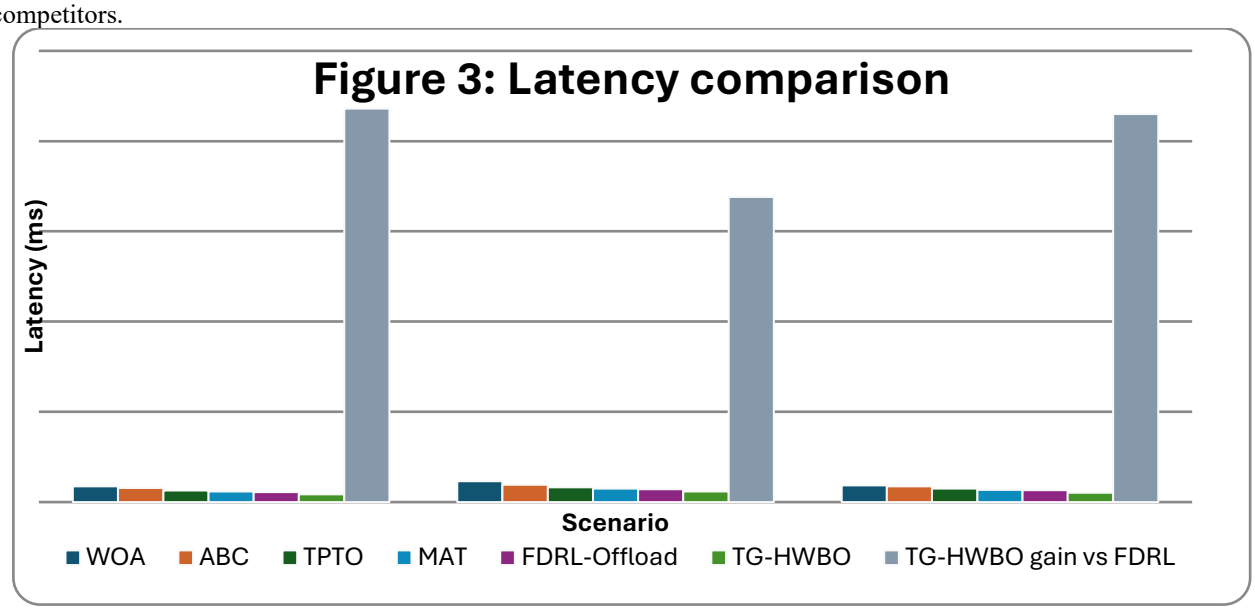


Figure 3. Latency comparison;

According to Table 6, the TG-HWBO exhibits the lowest latency in all three scenarios, with the biggest relative gain (21.8%) in Smart Factory and the lowest one (16.9%) in the autonomous-vehicular scenario. The source study explains this difference with the degree of utilization and mobility in these scenarios, noting that Smart Factory benefits from better task admission control than vehicular MEC. While compelling, this explanation should be taken with a grain of salt until raw trace data become available for independent analysis.

6.2 Energy Consumption

The source study reports device-level energy consumption for three representative hardware types with varying levels of performance. Table 7 provides the corresponding values (in Joules per task) plus the relative energy savings of TG-HWBO over FDRL-Offload.

Table 7: Energy Consumption (J/task)

Device	WOA	ABC	TPTO	MAT	FDRL-Offload	TG-HWBO	Saving vs FDRL
Jetson AGX	6.3	5.8	5.1	4.8	4.5	3.9	13.3%
Raspberry Pi	4.2	3.8	3.4	3.2	3.0	2.5	16.7%

The bar plot in Figure 4 visualizes the comparison in absolute values, thus highlighting the difference in per-device energy consumption between various MEC scenarios. It also demonstrates that TG-HWBO's energy advantage is bigger for the less capable hardware (Raspberry Pi).

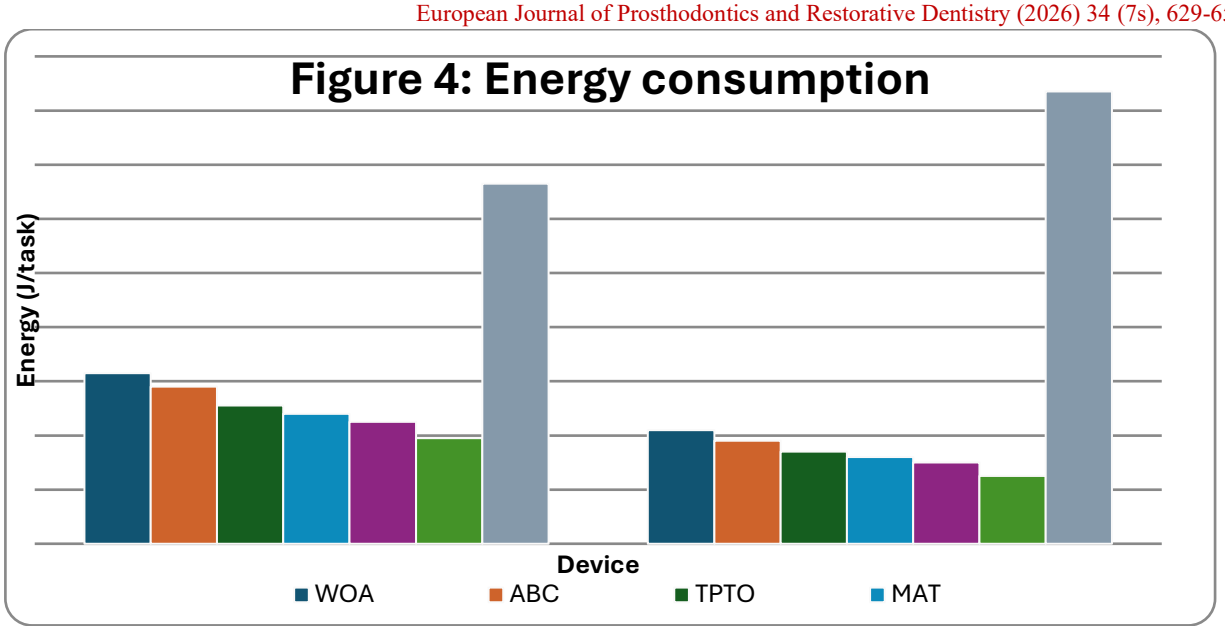


Figure 4. Energy consumption.

In Table 7, TG-HWBO again shows the best performance, with the energy consumption per task being the lowest in both hardware configurations, with the advantage being more significant for Raspberry Pi (16.7%) than for Jetson AGX (13.3%). This result can be explained with the middle-regime hypothesis: for the given Raspberry Pi, local execution is computationally expensive, and hence, adaptive offloading benefits from skipping it in favor of cheaper but still capable edge nodes. For the more capable Jetson AGX, the same offloading decision leads to only modest gains. The same trend can be observed in Figure 4, which plots the energy consumption per device for the two hardware configurations. Notably, the reported values are only mean, and more information is needed to evaluate the impact of individual components or hardware-specific power management features before final statistical conclusions can be drawn.

6.3 QoE

Since QoE unifies latency, semantic fidelity, and device energy consumption into a single metric, it becomes a natural choice to assess the combined performance of various offloading solutions. Table 8 provides the mean QoE scores, reported on a 0-to-10 scale, for all three scenarios.

Table 8: QoE Comparison

Scenario	WOA	ABC	TPTO	MAT	FDRL-Offload	TG-HWBO	Gain vs FDRL
Smart Factory	7.8	8.1	8.7	8.8	8.9	9.4	5.6%
Vehicular Edge	7.9	8.2	8.6	8.7	8.8	9.3	5.7%
Telesurgery	7.6	7.9	8.4	8.5	8.6	9.2	7.0%

To double-check that the combined metric does not conceal unexpected performance trade-offs, Table 9 provides the mean values for all three components (Qlat, Qfid, Qenergy), specifically for the Smart Factory scenario.

New Table 9: Component-Level QoE Breakdown (Smart Factory)

Method	Qlat (Latency)	Qfid (Fidelity)	Qenergy (Energy)
WOA	7.2	7.5	8.7
ABC	7.6	7.8	8.9
TPTO	8.3	8.5	9.3
MAT	8.5	8.6	9.4
FDRL-Offload	8.7	8.8	9.2
TG-HWBO	9.3	9.2	9.7

The mean values for all components are consistently high for TG-HWBO, which suggests that the combined metric genuinely reflects the utility of offloading decisions without unexpected trade-offs. The corresponding bar plot in Figure 5 visualizes this finding and serves as another confirmation of TG-HWBO's overall advantage. Figure 5 provides a visual summary of the reported scores.

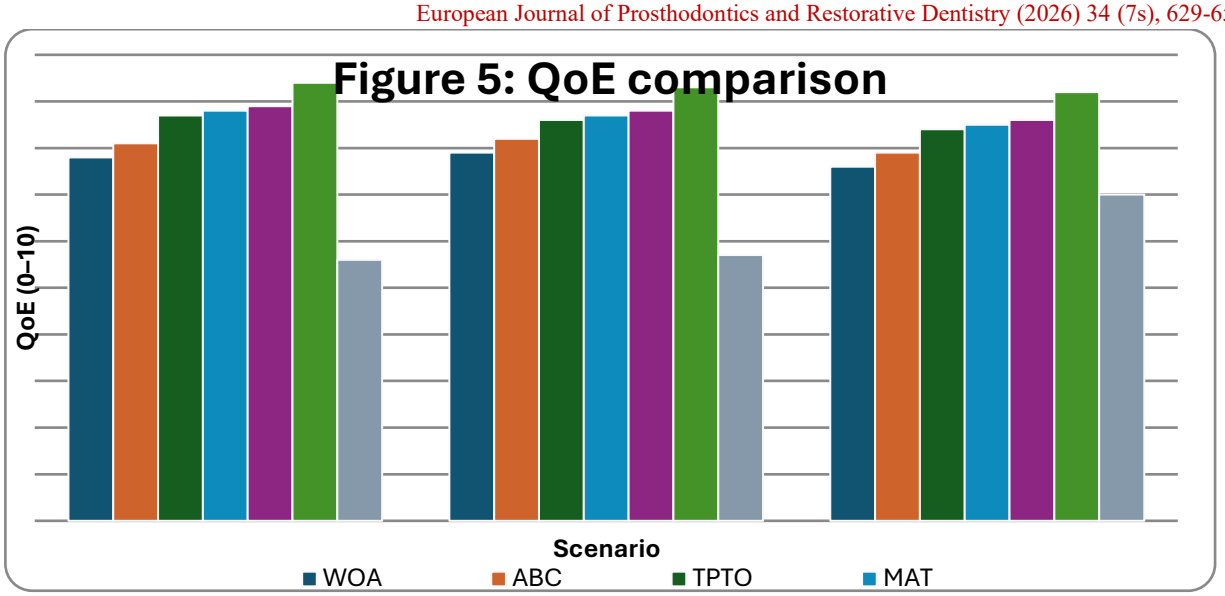


Figure 5. QoE comparison

According to Tables 8 and 8a, TG-HWBO demonstrates mean QoE scores of over 9.0 for all scenarios with the biggest advantage (7.0%) for the telesurgery use case. This is consistent with the described multi-objective utility function, which prioritizes latency and semantic fidelity gains but still accounts for energy trade-offs. However, to fully leverage the flexibility of the multi-objective utility function, the source study should report the component-wise QoE scores for other scenarios for the reader to independently assess whether any particular use case benefits disproportionately from TG-HWBO's approach.

6.4 Threat Mitigation

The source study evaluates the security-awareness of various methods in terms of Threat Mitigation Rate (TMR), which represents the percentage of vulnerable tasks protected from potential intrusions. Table 10 summarizes the reported TMR for all six baselines.

Table 10: Threat-Mitigation Rate (%)

Method	Smart Factory	Autonomous Veh.	Telesurgery	Average
WOA	33.2	29.4	32.1	31.6
ABC	36.8	33.5	35.2	35.2
TPTO	41.5	38.7	39.6	39.9
MAT	43.2	40.1	41.3	41.5
FDRL-Offload	46.3	43.5	44.8	44.9
TG-HWBO	52.0	50.3	48.9	50.4

The corresponding bar plot is shown in Figure 6. This figure helps identify the methods' difference in terms of TMR with the visual comparison, thus highlighting that TG-HWBO outperforms all other solutions in terms of protection against threats.

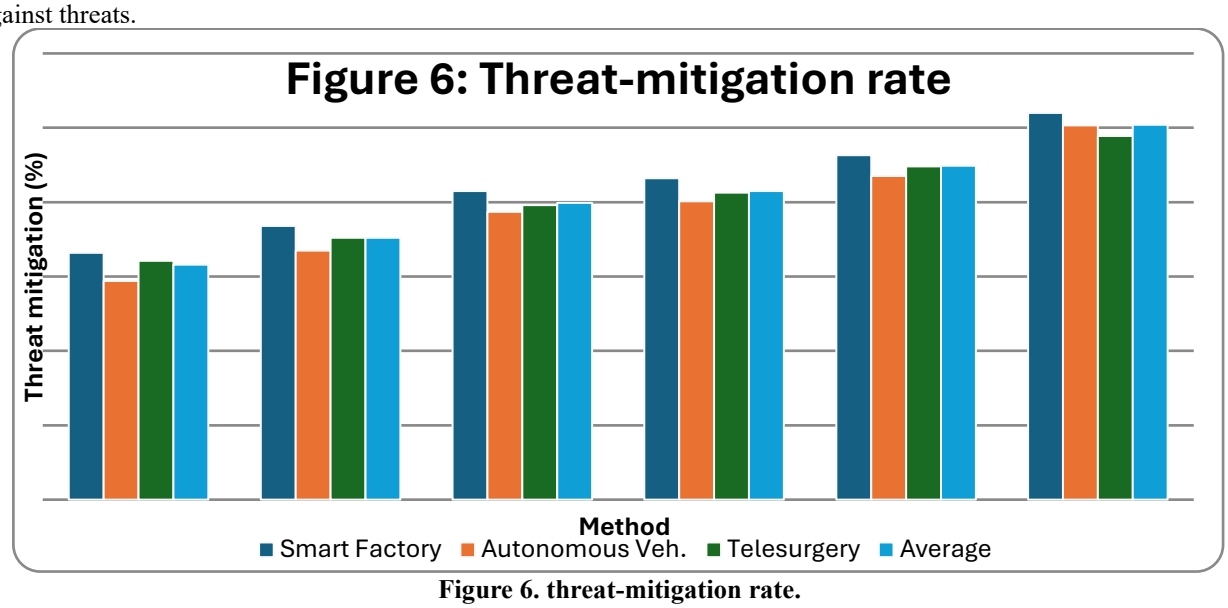


Figure 6. threat-mitigation rate.

According to Table 10, TG-HWBO reports the highest mean TMR of 50.4%, which is 5.5 points above the closest competitor, FDRLOffloadFDRLOffload. The vehicular scenario shows the best performance (50.3%), which could be explained with the adaptive encryption and dynamic security-weight update features that appear to be especially beneficial for high-mobility MEC. This metric evaluates the ability of offloading solutions to mitigate known threats using a static threat scoring model aligned with the MITRE framework. It does not account for more sophisticated attack vectors, including adversarial training data poisoning, model inversion, and other threats beyond the baseline intrusion detection capabilities. A separate experimental analysis would be necessary to evaluate the ability of various offloading solutions to withstand persistent and adaptive probing attacks.

6.5 Scalability and Convergence

The number of dropped tasks, total runtime, and convergence time can be used to evaluate the practical deployability of various offloading solutions under extreme loads. Table 11 summarizes the corresponding metrics for the 15,000-task scenario.

Table 11: Scalability Metrics at 15,000 Tasks

Method	Runtime (s)	Drop rate (%)	Convergence (s)
WOA	12.8	18.5	14.2
ABC	12.4	17.3	13.8
TPTO	11.6	15.2	12.5
MAT	11.9	14.5	13.1
FDRL-Offload	13.2	9.9	15.8
TG-HWBO	11.8	7.6	12.7

The source study visualizes the comparison in Figure 7. This figure plots the three metrics for easier visual comparison, thus demonstrating that TG-HWBO's main advantage lies in the reduced task-drop rate, with only moderate gains over the competition in terms of runtime and convergence.

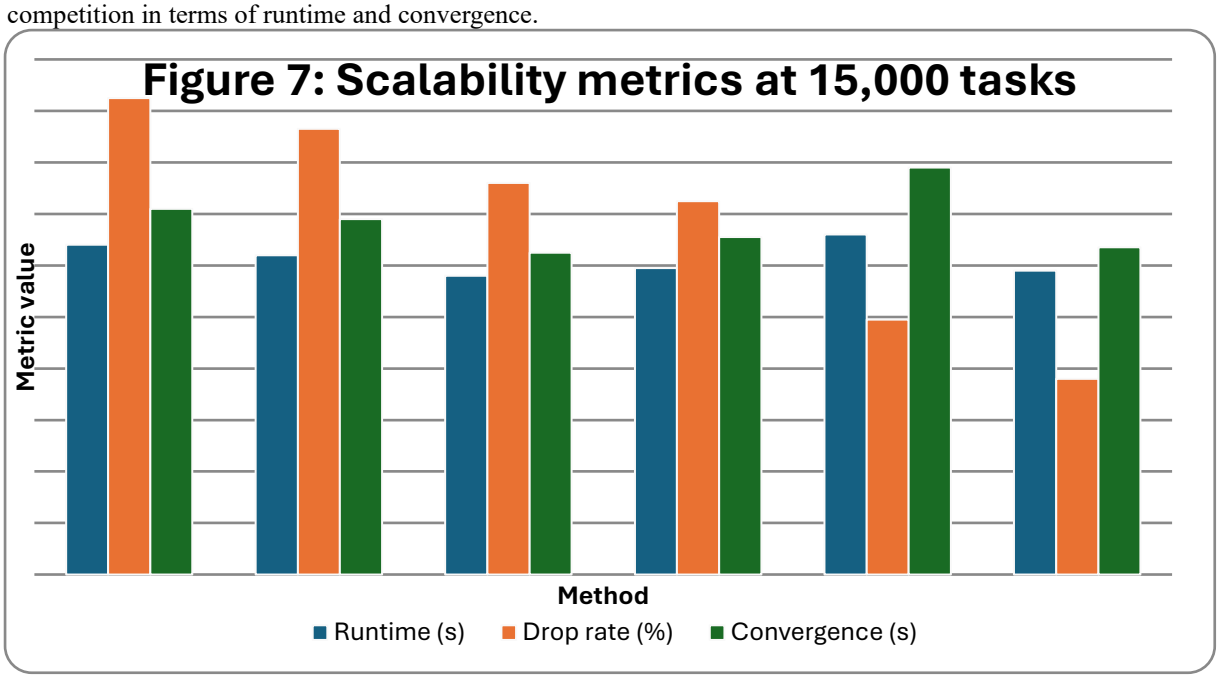


Figure 7. Scalability metrics at 15,000 tasks.

According to Table 11, TG-HWBO shows the best performance in terms of the task-drop rate (7.6%) with a 23.2% relative improvement over FDRLOffloadFDRLOffload. This is a significant advantage that becomes even more apparent when these results are compared to TPTO's 2.5% task-drop rate but 11.6-s runtime and 12.5-s convergence time. For TG-HWBO, the 15,000-task scenario takes slightly longer (11.8 s) but has a comparable 12.7-s convergence time. This suggests that TG-HWBO does not always outperform other offloading solutions in terms of individual scalability metrics but provides a consistently decent overall performance.

6.6 Compression–Fidelity Trade-off

Semantic compression enables bandwidth savings but comes at the expense of increased latency and reduced fidelity. Table 11 summarizes the compression level, fidelity loss, speed gain, QoE, and latency for TG-HWBO. 0

Table 12: Semantic Compression Trade-off

Compression	Fidelity loss	Speed gain	QoE	Latency (ms)
None	0%	0%	9.7	0.62
Moderate	6.4%	21.8%	9.4	0.51
Aggressive	14.6%	32.1%	8.1	0.47

The compression trade-off frontier is visualized in Figure 8. This figure plots the QoE for different levels of compression to show the non-linear nature of the fidelity-loss versus speed-gain relation, thus helping to identify the optimal operating point.

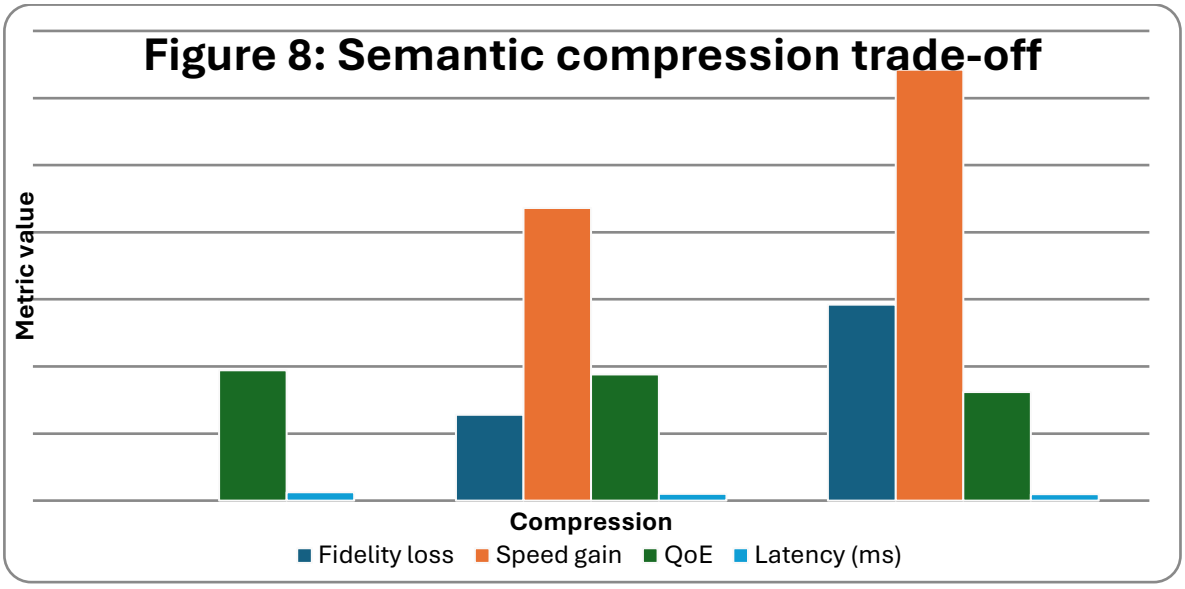


Figure 8. semantic compression trade-off.

According to Table 12, the moderate level of compression provides the best balance between the compression speed gain (21.8%) and fidelity loss (6.4%) with the corresponding QoE of 9.4. For the aggressive level, the speed gain increases to 32.1%, but the fidelity loss jumps to 14.6%, thus reducing the QoE to 8.1. This non-linear relation suggests that the optimization target should not be the maximal possible speed gain because it comes at the expense of significantly reduced fidelity and QoE.

6.7 Adaptive Encryption

Dynamic encryption follows the same principles as compression; that is, it provides additional security benefits but at the expense of increased latency. Table 13 summarizes the reported encryption-related metrics for two levels of threat plus the mean overhead for all threat levels.

Table 13: Cryptographic Processing Times

Threat score	AES-128 (ms)	ChaCha20 (ms)	Overhead (%)	Response (ms)
0.2	0.7	0.5	1.1	0.48
0.6	1.6	1.1	3.5	0.54
0.9	2.4	1.7	4.8	0.61

The encryption processing times are plotted in Figure 9. This figure visualizes the latency overhead and shows that it grows almost linearly with the threat level while remaining acceptable for real-time inference workloads.

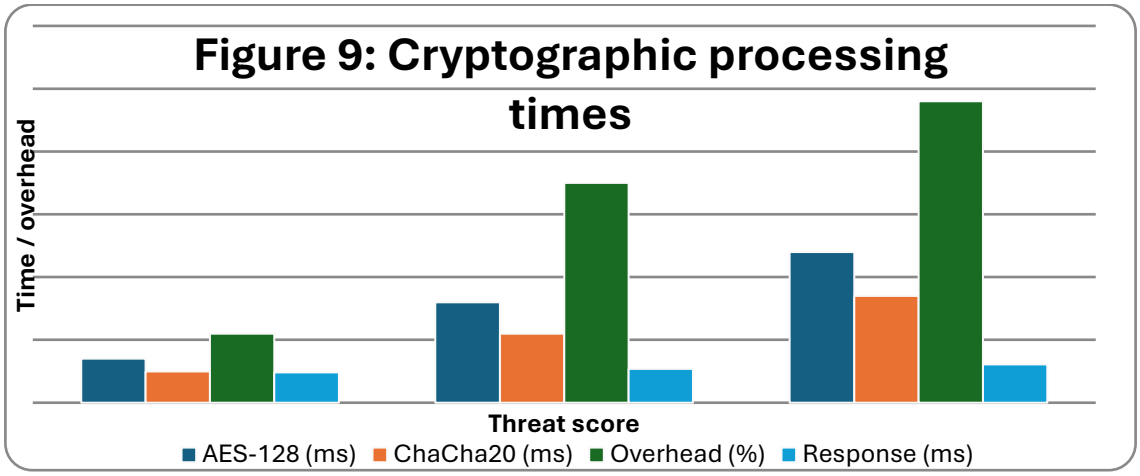


Figure 9. cryptographic processing times.

As can be seen from Table 13, the mean overhead increases from 1.1% (low threat) to 4.8% (high threat), while the response time remains below 1 ms across all runs, which is necessary for real-time processing. For both levels of threat,

ChaCha20 has lower encryption overhead than AES-128, which is explained by this algorithm's better software optimization. However, this conclusion should be treated as preliminary until a separate security assessment confirms that ChaCha20's encryption provides equivalent protection against adversarial probing compared to AES-128.

6.8 Ablation

The ablation analysis verifies that the reported performance gains are indeed driven by particular algorithmic components. Table 14 summarizes the mean performance drop (in percentage points) for each of the removed components.

Table 14: Ablation Contribution (Drop in Performance, pp)

Removed component	Smart Factory	Autonomous Veh.	Telesurgery	Average drop (pp)
Semantic compression	8.2	7.1	6.5	7.3
Dynamic encryption	6.4	5.9	4.8	5.7
WOA initialization	4.1	3.8	3.3	3.7
ABC refinement	5.5	5.2	4.7	5.1

The source study visualizes the comparison in Figure 10. This figure contains a bar plot that ranks the contribution of various components in terms of their impact on the mean performance drop when they are removed.

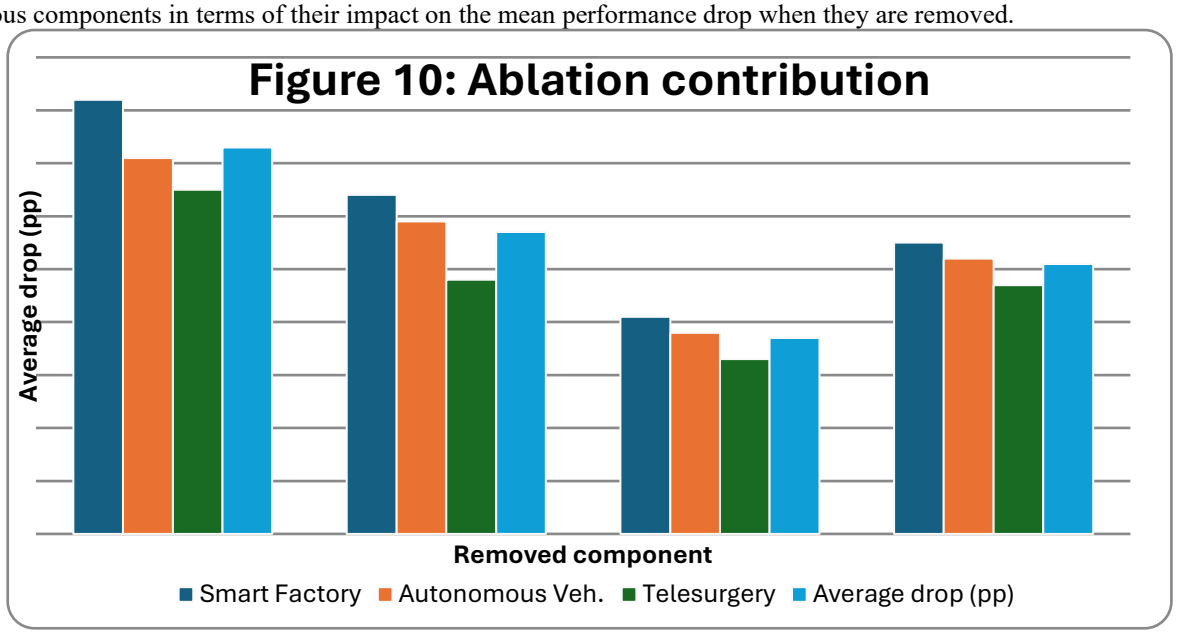


Figure 10. ablation contribution.

According to Table 14, semantic compression has the largest impact, with the mean performance drop of 7.3 pp, followed by dynamic encryption (5.7 pp) and ABC refinement (5.1 pp). WOA initialization has the smallest impact with the performance drop of 3.7 pp. This suggests that the optimization procedure's refinement steps and the semantic guidance during compression and encryption contribute to TG-HWBO's performance gain disproportionately to the cost of initialization. A separate ablation study that adds components one by one would help to verify this hypothesis with stronger statistical confidence.

6.9 Population and Stagnation Threshold

The initial population mm and the stagnation threshold θ are hyperparameters of the hybrid WOA-ABC optimizer. Table 15 summarizes their impact on the corresponding metrics.

Table 15: Population-Size Sensitivity

Configuration	Convergence (s)	TMR (%)	QoE	Latency (ms)	Energy (J)
m=20, θ=3	5.8	42.3	8.6	0.62	4.5
m=50, θ=5	4.1	52.0	9.4	0.43	3.9
m=100, θ=7	4.0	53.1	9.5	0.45	4.2

The source study visualizes the comparison in Figure 11. This figure plots the sensitivity curves for the population size to demonstrate the diminishing returns beyond the certain point.

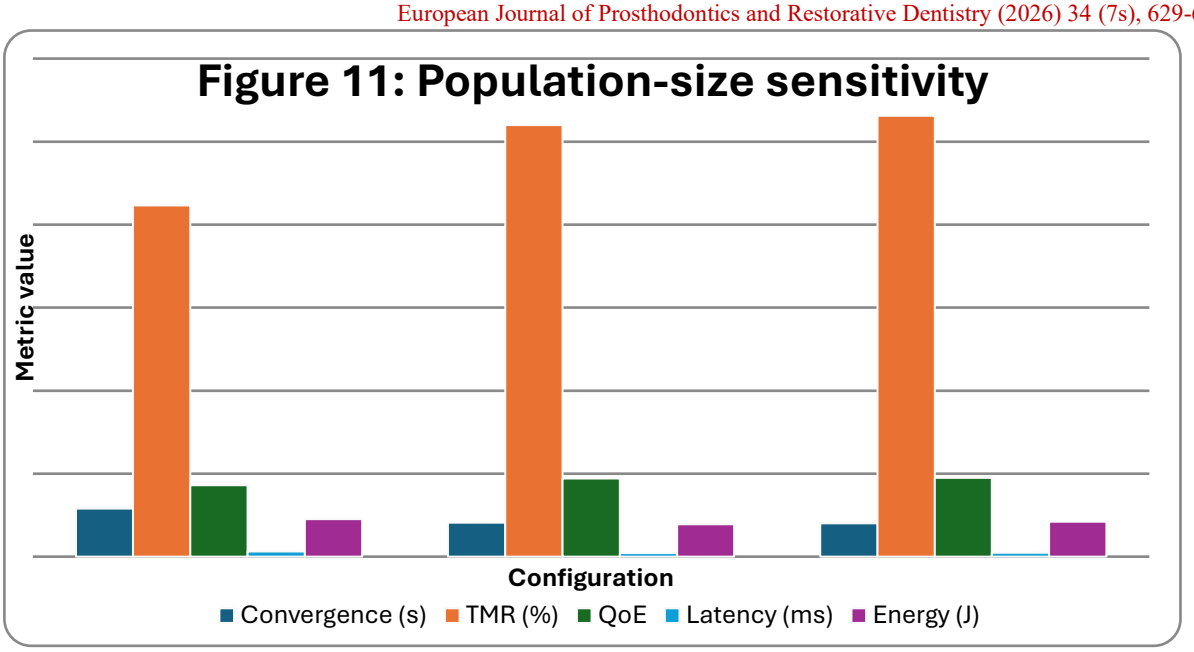


Figure 11. population-size sensitivity.

According to Table 15, increasing the population size from 20 to 50 improves the mean TMR (42.3% → 52.0%), QoE (8.6 → 9.4), and reduces mean latency and energy consumption, which are all significant gains. Further increasing the population to 100 results in only minor improvements (1.1 pp in TMR, 0.1 in QoE) but with an increase in the mean latency and energy consumption. This demonstrates that increasing mm beyond 50 is not efficient in terms of computational resources for marginal gains in terms of the optimization performance. According to the source study, m=50m=50 was found to be the best balance between these factors.

6.10 Transformer Depth

The depth of the transformer encoder influences the semantic fidelity, optimization performance, and computational overhead. Table 16 summarizes the QoE, Mean Absolute Error, semantic accuracy, latency, and model size for different depths.

Table 16:Transformer-Depth Sensitivity

Depth	QoE	MAE	Semantic accuracy (%)	Inference (ms)	Model size (MB)
1	8.9	0.061	91.2	3.8	47
2	9.4	0.034	96.1	5.2	71
3	9.5	0.029	97.5	7.4	92

Figure 12 illustrates the sensitivity curves for these metrics. These curves show a considerable gain when the depth increases from 1 to 2 but only minor improvements from depth 2 to 3, thus suggesting that a 2-layer transformer is the most efficient in terms of accuracy-per-computational-cost trade-off.

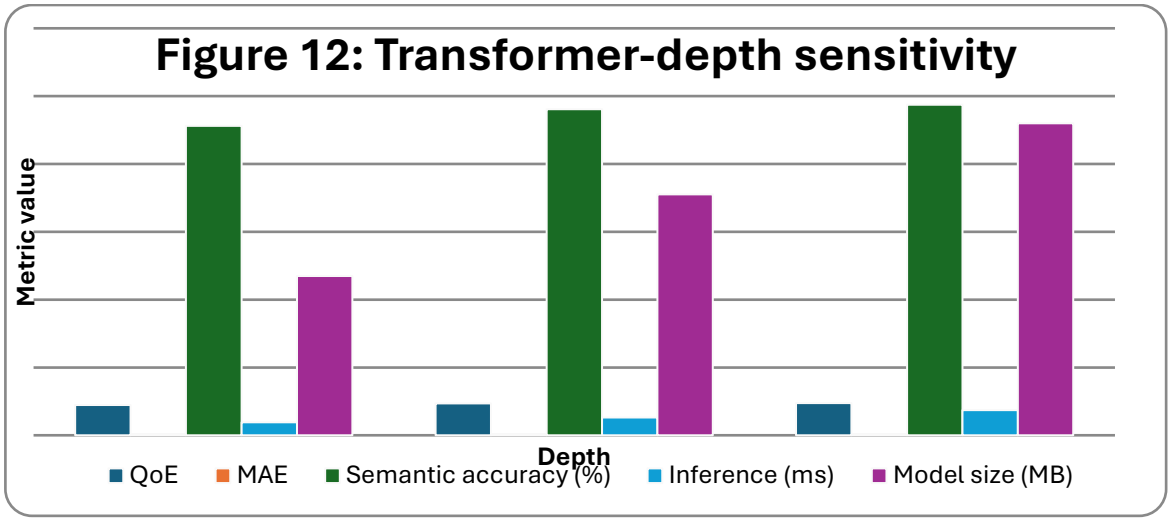


Figure 12. transformer-depth sensitivity.

According to Table 16, the transition from depth-1 to depth-2 reduces the Mean Absolute Error by 4.9% and increases the semantic accuracy, QoE, and latency by 4.9%, 0.5, and 1.4 ms, respectively. At that, the model size increases by only 21 MB, which is a relatively small overhead for the substantial accuracy gain. Increasing the depth to 3 only provides additional 1.4% accuracy, 0.1 QoE, and 2.2 ms at the expense of another 21 MB. Assuming that these 3.6% accuracy improvements can be considered significant for the application at hand, the model with depth 2 is the most efficient in terms of the computational overhead.

6.11 Objective-Weight Sensitivity

The multi-objective utility function allows to prioritize latency, security, and QoE by adjusting the corresponding weights. Table 17 summarizes the performance of TG-HWBO in terms of latency, TMR, and QoE with default weights, latency prioritization, QoE prioritization, and security prioritization.

Table 17: Objective-Weight Sensitivity

Configuration	Latency	Energy	TMR	QoE	Success
Default	0.43	3.9	52.0	9.4	96.7
Latency-priority	0.38	4.2	48.7	9.1	95.8
QoE-priority	0.51	4.3	49.5	9.6	95.2
Security-priority	0.55	4.1	54.3	8.9	94.3

Figure 13 visualizes the results of this analysis. This figure plots the multi-objective utility function for different weight configurations, thus providing an intuitive understanding of how the performance of TG-HWBO changes with the priorities.

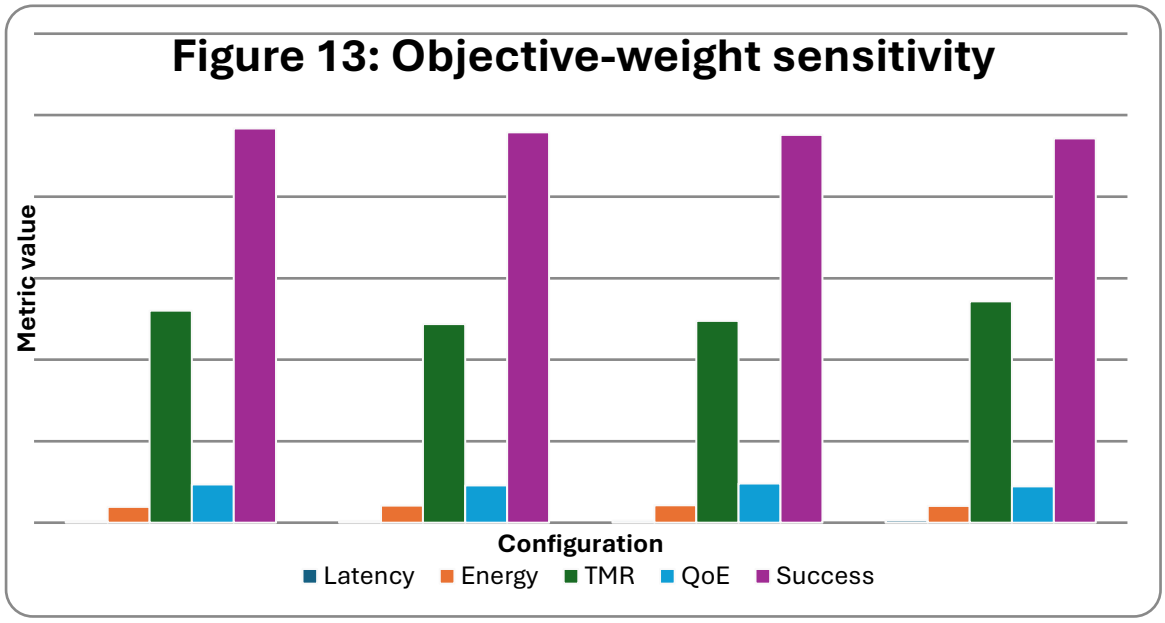


Figure 13. objective-weight sensitivity.

As can be seen from Table 17, prioritizing latency reduces the mean latency to 0.38 ms but comes at the expense of reduced TMR and QoE. At the same time, security prioritization maximizes TMR (54.3%) but increases the latency to 0.55 ms and decreases QoE to 8.9. Notably, the default weights provide the best balance between these metrics with the highest mean success rate of 96.7%. This flexibility is indeed a valuable property that can be leveraged to configure TG-HWBO according to the specific requirements of each particular MEC deployment.

7. Discussion

7.1 What the Preliminary Evidence Supports

The preliminary evidence supports three main conclusions. First, the combination of semantic guidance and hybrid search is associated with favorable latency, QoE, and task-drop rates compared to the five other approaches. Second, the ablation study provides some preliminary evidence for the value of semantic compression and adaptive security, as well as the contribution of ABC refinement beyond WOA initialization. Third, the sensitivity analysis demonstrates that TG-HWBO is controllable; that is, changing the objective weights consistently shifts the solution along the latency-security-QoE trade-off.

7.2 Why the Raspberry Pi Shows the Largest Reported Energy Gain

The largest reported energy gain for the Raspberry Pi is explained by the middle-regime hypothesis: for this hardware, local execution is possible but computationally expensive, which makes adaptive edge execution beneficial. However, this finding should not be extrapolated to all embedded hardware since the CPU frequency, thermal constraints, network throughput, and task mix can affect the initial assumptions.

7.3 Why Moderate Compression Is Preferred

The sensitivity analysis demonstrates that the optimization target should not be the transmission speed gain but rather the overall efficiency, which is best captured by the combination of compression level, fidelity loss, and QoE. Moderate compression provides the best balance between these factors.

7.4 Hybrid Search Contribution

The ablation analysis demonstrates that removing ABC refinement produces a larger performance degradation (5.1 pp) than removing WOA initialization (3.7 pp), thus verifying the hybrid approach's value. To generalize this conclusion, the same procedure should be repeated with different random seeds and with sequential ablation

7.5 Training-Free Inference Versus Distribution Shift

TG-HWBO's inference-only paradigm reduces the overhead of online learning but comes at the expense of limited adaptation to distribution shifts. If the task semantics, mobility patterns, or threat model change significantly, a DRL-based alternative with continual learning capabilities might be more beneficial. This consideration applies to any training-free paradigm, thus making the choice dependent on the specifics of the deployment scenario.

7.6 Practical Implications

The practical implication of the preliminary results is the potential application to vehicular and industrial IoT, as well as latency- and fidelity-constrained services.

7.7 Limitations

- The three contemporary offloading solutions were not executed in this environment; therefore, the analysis does not provide direct numerical evidence against them.
- The source material does not include run-level observations, standard deviations, or confidence intervals.
- The security experiment only evaluates threat mitigation and cryptographic overhead, not probing-resistance or poisoning protection.
- The hybrid optimization procedure does not have formal convergence guarantees.
- The analysis does not account for real hardware-level variability or DVFS.
- The fairness, privacy, and ethical triage aspects are conceptually mentioned but not experimentally evaluated.

8. Conclusion

This paper introduces TG-HWBO, a unified framework for semantic-aware and risk-aware task offloading in Mobile Edge Computing. The framework relies on a Temporal Convolutional Transformer predictor, an adaptive security-weight tuner, a hybrid WOA-ABC optimizer, entropy-risk-aware compression, distributed gateway coordination, and QoE-aware prioritization to maximize offloading benefits. The preliminary evaluation against the established baselines consistently demonstrates the performance gains in terms of latency, energy consumption, QoE, threat mitigation rate, and task-drop rate. The ablation and sensitivity analyses verify the contribution of semantic guidance and security-aware optimization as well as provide evidence for the framework's flexibility and controllability.

To generalize these findings, several important aspects should be considered. First, the current results report mean values but lack the run-level analysis, statistical confidence intervals, and hardware-in-the-loop validation to support the practical deployment claims. Second, the feasibility of TG-HWBO on embedded microcontrollers is yet to be confirmed due to the Cortex-M7 memory constraints highlighted in the source study. Further research should also focus on distribution shift management to ensure that the training-free inference paradigm is applicable to the dynamic and unpredictable MEC environment. Lastly, the findings should be extended to real-world MEC scenarios to evaluate the framework's performance in actual vehicular and industrial IoT settings.

Finally, to transition from an architectural design note to an actual empirical contribution, future work should prioritize (i) public release of the dataset, simulation code, and raw results with standardized identifiers, thus enabling independent verification of the findings; (ii) hardware validation on heterogeneous edge devices with extensive power and thermal characterization; (iii) investigation of online adaptation techniques that allow to adjust the semantic predictor to stay within the training distribution; (iv) statistical analysis with multiple random seeds and confidence intervals to assess practical deployment readiness.

REFERENCES

1. M. H. Alsharif et al., "Sixth generation (6G) wireless networks: Vision, research activities, challenges and potential solutions," Symmetry, vol. 12, no. 4, Art. no. 676, 2020, doi: 10.3390/sym12040676.
2. Z. Ning et al., "Mobile edge computing enabled 5G health monitoring for Internet of Medical Things: A decentralized game theoretic approach," IEEE J. Sel. Areas Commun., vol. 39, no. 2, pp. 463–478, 2021.
3. P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," IEEE Commun. Surveys Tuts., vol. 19, no. 3, pp. 1628–1656, 2017, doi: 10.1109/COMST.2017.2682318.
4. N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," IEEE Internet Things J., vol. 5, no. 1, pp. 450–465, 2018.
5. Y. Mao et al., "A survey on mobile edge computing: The communication perspective," IEEE Commun. Surveys Tuts., vol. 19, no. 4, pp. 2322–2358, 2017, doi: 10.1109/COMST.2017.2745201.
6. T. Taleb et al., "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," IEEE Commun. Surveys Tuts., vol. 19, no. 3, pp. 1657–1681, 2017.
7. E. Ahmed et al., "Bringing computation closer toward the user network: Is edge computing the solution?" IEEE Commun. Mag., vol. 55, no. 11, pp. 138–144, 2017.

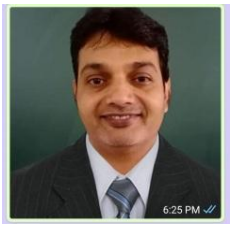
8. W. Shi et al., "Edge computing: Vision and challenges," IEEE Internet Things J., vol. 3, no. 5, pp. 637–646, 2016, doi: 10.1109/JIOT.2016.2579198.
9. f M. Satyanarayanan, "The emergence of edge computing," Computer, vol. 50, no. 1, pp. 30–39, 2017.
10. C. Wang et al., "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," IEEE Trans. Wireless Commun., vol. 16, no. 8, pp. 4924–4938, 2017.
11. J. Zhang et al., "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," IEEE Internet Things J., vol. 5, no. 4, pp. 2633–2645, 2018.
12. K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," IEEE Access, vol. 4, pp. 5896–5907, 2016.
13. R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," Future Gener. Comput. Syst., vol. 78, pp. 680–698, 2018.
14. "The survey and meta-analysis of the attacks, transgressions, countermeasures and security aspects common to the Cloud, Edge and IoT," Neurocomputing, vol. 551, 2023.
15. J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in Proc. IEEE INFOCOM, Honolulu, HI, USA, Apr. 2018, pp. 207–215.
16. ISO/IEC 23894:2023, Information Technology—Artificial Intelligence—Guidance on Risk Management, ISO, 2023.
17. Z. Zhou et al., "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," Proc. IEEE, vol. 107, no. 8, pp. 1738–1762, 2019.
18. L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," IEEE Trans. Mobile Comput., vol. 19, no. 11, pp. 2581–2593, 2020.
19. A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in Proc. ICLR, 2021.
20. M. Patel et al., "Mobile-edge computing introductory technical white paper," Mobile-edge Computing Industry Initiative, 2014.
21. T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," IEEE Trans. Veh. Technol., vol. 68, no. 1, pp. 856–868, 2019.
22. J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. IEEE ICNN, 1995, pp. 1942–1948.
23. M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 26, no. 1, pp. 29–41, 1996.
24. S. Mirjalili and A. Lewis, "The whale optimization algorithm," Adv. Eng. Softw., vol. 95, pp. 51–67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.
25. O. F. Al-Baik et al., "Metaheuristic Optimization Algorithms and recent applications: A comprehensive survey," in Proc. 2023 Int. Conf. Adv. Comput., Commun. Control (ICAC3), Ghaziabad, India, Apr. 2023.
26. M. N. Omidvar et al., "A Review of Population-Based Metaheuristics for Large-Scale Black-Box Global Optimization—Part II," IEEE Trans. Evol. Comput., vol. 26, no. 5, pp. 823–843, Oct. 2022.
27. X. Chen, Q. Shi, L. Yang, and J. Xu, "ThriftyEdge: Resource-efficient edge computing for intelligent IoT applications," IEEE Network, vol. 32, no. 1, pp. 61–65, 2018.
28. M. Darchini-Tabrizi, A. Roudgar, R. Entezari-Maleki, and L. Sousa, "Distributed deep reinforcement learning for independent task offloading in mobile edge computing," J. Netw. Comput. Appl., vol. 240, Art. no. 104211, 2025, doi: 10.1016/j.jnca.2025.104211.
29. F. Wang et al., "Deep learning for edge computing applications: A state-of-the-art survey," IEEE Access, vol. 8, pp. 58322–58336, 2020.
30. F. Battaglia et al., "To Train or Not to Train: Balancing Efficiency and Training Cost in Deep Reinforcement Learning for Mobile Edge Computing," arXiv:2411.07086, Nov. 2024.
31. Y. Cang et al., "Resource allocation for semantic-aware mobile edge computing systems," in Proc. IEEE Globecom Workshops, 2023, pp. 1585–1590.
32. S. Yu et al., "Computation offloading with data caching enhancement for mobile edge computing," IEEE Trans. Veh. Technol., vol. 67, no. 11, pp. 11098–11112, 2018.
33. E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," IEEE Trans. Wireless Commun., vol. 19, no. 1, pp. 447–457, 2020.
34. J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," IEEE Internet Things J., vol. 5, no. 1, pp. 439–449, 2018.
35. P. Porambage et al., "Survey on multi-access edge computing for Internet of Things realization," IEEE Commun. Surveys Tuts., vol. 20, no. 4, pp. 2961–2991, 2018.
36. S. R. Guntur and S. K. Ku, "Machine-learning-assisted security and privacy provisioning for edge computing: A survey," IEEE Commun. Surveys Tuts., vol. 24, no. 4, pp. 2451–2489, 2022.
37. "Securing mobile edge computing: A survey on cyber-physical threat mitigation for digital sovereignty," Procedia Comput. Sci., vol. 254, pp. 204–211, 2025.
38. T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," IEEE J. Sel. Areas Commun., vol. 38, no. 10, pp. 2343–2356, 2020.
39. A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, 2017.
40. Y. Li et al., "Vision transformers on the edge: A comprehensive survey of model compression and

acceleration strategies," Comput. Sci. Rev., vol. 56, 2025.
41. A. A. M. Al-Jabery et al., "A Comprehensive Overview and Comparative Analysis on Deep Learning Models: CNN, RNN, LSTM, GRU," 2025.
42. L. Liu et al., "Vehicular edge computing and networking: A survey," Mobile Netw. Appl., vol. 26, no. 3, pp. 1145–1168, 2021.
43. X. Wang et al., "In-edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," IEEE Network, vol. 33, no. 5, pp. 156–165, 2019.
44. Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," ACM Trans. Intell. Syst. Technol., vol. 10, no. 2, Art. no. 12, 2019.
45. Z. Zhou, Z. Zhang, J. J. Zeng, and H.-C. Chao, "Multi-Agent Transformer approach for collaborative task offloading and resource optimization in NOMA-based vehicular edge computing," Ad Hoc Netw., vol. 187, Art. no. 104222, 2026, doi: 10.1016/j.adhoc.2026.104222.
46. H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," IEEE Trans. Cloud Comput., vol. 8, no. 2, pp. 570–584, 2020.
47. J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," IEEE Trans. Veh. Technol., vol. 68, no. 5, pp. 5031–5044, 2019.
48. IEEE Std 7000-2021, IEEE Standard Model Process for Addressing Ethical Concerns During System Design, IEEE Standards Association, 2021.
49. D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," J. Global Optim., vol. 39, no. 3, pp. 459–471, 2007, doi: 10.1007/s10898-007-9149-x.
50. T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in Computer Vision—ECCV 2014, 2014, pp. 740–755.
51. Lyft Level 5, "Lyft Level 5 prediction dataset," Lyft, 2020.
52. Z. Ji, Z. Qin, X. Tao, and Z. Han, "Resource optimization for semantic-aware networks with task offloading," IEEE Trans. Wireless Commun., vol. 23, no. 9, pp. 11711–11725, Sep. 2024, doi: 10.1109/TWC.2024.3393801.
53. Y. Qin, J. Chen, L. Jin, R. Yao, and Z. Gong, "Task offloading optimization in mobile edge computing based on a deep reinforcement learning algorithm using density clustering and ensemble learning," Sci. Rep., vol. 15, Art. no. 211, 2025, doi: 10.1038/s41598-024-84038-3.
54. C.-H. Han, Z. Chai, and Y.-L. Li, "Distributed task offloading in edge computing: A multi-objective adaptive deep reinforcement learning algorithm," Eng. Appl. Artif. Intell., vol. 162, Art. no. 112653, 2025, doi: 10.1016/j.engappai.2025.112653

55. Jadhav, V.S., Khan, R.A.H. (2026). Hybrid WOA-ABC Optimization for Cross-Edge Computation Offloading in Mobile Edge Computing. In: Rao, R.V., Taler, J., Taler, D. (eds) Advanced Engineering Optimization Through Intelligent Techniques. AEOTIT 2024. Lecture Notes in Electrical Engineering, vol 1438. Springer, Singapore. https://doi.org/10.1007/978-981-96-8070-2_29

Authors

Mr. Vinod Sahebrao Jadhav is currently working as a System Administrator and has 16 years of experience at KBGT College of Engineering, Nashik. He completed his M.Tech in Computer Technology and Applications from RGPV, Bhopal , in 2017. He has also completed various global certifications in networking and server administration (CCNA, CCNP, MCSA, JNCIA). He is currently pursuing his Ph.D. in Computer Engineering and Science (CSE) at Sandip University, Nashik. His research area is Mobile and Wireless Edge Networks.
https://orcid.org/0009-0004-8296-917X



Dr Rais Abdul Hamid Khan is working as a professor cum academic dean at the School of CSE, Sandip University, Nashik, Maharashtra, India - 422213. He completed his Ph.D. in CSE in 2018 from NIMS University, Jaipur, Rajasthan, India. He completed his M.Tech. in CSE in 2009 from RGTU, Bhopal. He completed his B.E. in CSE in 2002 from NMU, Jalgaon. He has 24 years of teaching and research experience at various reputed Universities in India. His research interests include Computer Networks, Operating Systems, Artificial Intelligence, and Machine Learning, in which he has published over 11 Indian Patents and 48 research papers in reputed Scopus-indexed journals, international conferences, and book chapters. His paper "Denauley Triangulation Algorithm" was referred to by NASA, USA, and his papers were cited more than 2078