

Paper Type: Original Article

Collaborative IoT Cloud and Edge Computing Solutions for Smart Cities

Amin Maghsoudi¹ , Shubham Hamirwasia^{2,*} 

¹ Department of Computer and Information Technology Engineering, Qa. c., Islamic Azad University, Qazvin, Iran; maghsoudi@iaui.ac.ir.

² Department of Computer Science, School of Engineering, KIIT University, Bhubaneswar, India; 22053109@kiit.ac.in.

Citation:

Received: 22 January 2026

Revised: 19 March 2026

Accepted: 12 May 2026

Maghsoudi, A., & Hamirwasia, S. (2026). Collaborative IoT cloud and edge computing solutions for smart cities. *Smart Internet of Things*, 3(3), 109-117.

Abstract

Smart-city Internet of Things (IoT) applications combine large device populations, heterogeneous workloads, intermittent wireless links, and demanding latency targets. Central cloud computing offers elastic capacity for computation-intensive analytics, whereas multi-access edge computing places compute and storage near data sources to reduce response time and backhaul traffic. Neither layer is universally optimal. This paper develops an expanded collaborative edge-cloud framework in which each task can be executed locally, at an edge server, or in a central cloud. A mathematical model captures communication delay, execution delay, device energy, monetary cost, server capacity, deadline compliance, and privacy restrictions. The resulting mixed-integer optimization problem is coupled with a distributed Deep-learning-Driven Task-Offloading (DDTO) policy that approximates high-quality decisions using locally observed system states and coordinated resource allocation. The manuscript also distinguishes proposed methodology from reported evidence: the frequently cited performance gain of up to 63% belongs to the experimental study by Wu et al. and is used here as prior evidence rather than as a new result. Security, interoperability, fairness, resilience, and governance are integrated into the design discussion. The framework provides a reproducible basis for future simulation or deployment studies in traffic control, video analytics, environmental sensing, emergency response, and smart energy management.

Keywords: Smart city IoT, Multi-access edge computing, Mobile cloud computing, Task offloading, Distributed deep learning, Resource allocation, Latency-energy optimization.

1 | Introduction

Urban systems increasingly depend on continuous sensing and data-driven control. The 2018 Revision of the United Nations World Urbanization Prospects reported that 55% of the global population lived in urban areas and projected this share to reach 68% by 2050. This concentration increases pressure on transportation,

energy, public safety, health, water, and environmental services. Smart-city platforms respond by connecting cameras, meters, vehicles, actuators, gateways, and citizen-facing applications to computational services.

IoT devices are typically constrained by battery capacity, processor speed, memory, and thermal limits. At the same time, modern workloads such as object detection, route optimization, anomaly detection, digital twins, and augmented-reality guidance may require substantial and predictable computation. Sending every task to a remote cloud can overload the backhaul and introduce wide-area-network delay. Processing every task locally can violate deadlines and drain device batteries. Edge computing occupies the middle layer by offering nearby shared resources, but edge nodes have finite capacity and can become congested [1].

The central research question is therefore not whether cloud or edge computing is better, but how a system should select among local, edge, and central-cloud execution under changing networking and workload conditions. The selection must account for task size, required CPU cycles, deadlines, wireless rates, server loads, energy budgets, monetary costs, and data sensitivity. A collaborative architecture can exploit the complementary strengths of all three locations.

This manuscript makes four contributions. First, it reorganizes the original discussion into an academically coherent system model. Second, it formulates task offloading as a constrained multi-objective optimization problem. Third, it presents a DDTO-oriented solution workflow and clarifies training, inference, coordination, and constraint enforcement. Fourth, it expands the analysis of security, interoperability, resilience, fairness, governance, and experimental design. No new experimental dataset is claimed; numerical performance values from prior work are explicitly attributed [2].

1.1 | Smart-City Application Requirements

Table 1. Representative workload tendencies in a collaborative smart-city architecture.

Application	Typical Workload Characteristic	Dominant Requirement	Preferred Tendency
Traffic video analytics	High input data and GPU-friendly inference	Low latency and high throughput	Edge first; cloud for batch learning
Environmental sensing	Small periodic messages	Energy efficiency and scalability	Local filtering plus cloud aggregation
Emergency response	Bursty, safety-critical events	Deadline reliability and resilience	Replicated edge with cloud backup
Smart grid monitoring	Continuous telemetry and control	Integrity, availability, privacy	Edge control plus central optimization
Urban digital twin	Large historical and streaming datasets	Compute capacity and consistency	Hierarchical edge-cloud pipeline

2 | Literature Review

2.1 | IoT in Smart Cities

IoT supplies the perception and actuation layer of a smart city. Atzori, Iera, and Morabito described IoT as an ecosystem of identifiable objects connected through communication capabilities and services. In urban deployments, heterogeneous sensors report physical conditions while actuators modify infrastructure states. The value of this layer depends not only on device connectivity but also on semantic interoperability, time synchronization, data quality, lifecycle management, and secure identity.

2.2 | Cloud Computing for Data Management

Cloud computing provides shared pools of compute, storage, databases, analytics, and machine-learning services. It is well suited to long-horizon optimization, model training, archival storage, cross-district correlation, and city-wide digital twins. Elastic scaling can absorb large workloads, but remote execution introduces upload delay, backbone traffic, and dependence on network availability. Data-governance requirements may also limit transfer of identifiable or safety-sensitive data beyond a local domain [3], [4].

2.3 | Edge Computing for Real-Time Processing

Edge computing moves services toward access networks and data sources. Shi et al. emphasized that computation at the network edge can support low-latency services and reduce unnecessary data movement. The European Telecommunications Standards Institute describes MEC as an IT service environment and cloud-computing capability at the network edge, with low latency, high bandwidth, and access to local network information. These properties are valuable for vehicle-to-infrastructure communication, video analytics, local hazard detection, and closed-loop control [5].

2.4 | Collaborative Edge-Cloud Frameworks

Collaboration allows preliminary filtering, inference, caching, and control to occur at the edge, while expensive training, global coordination, and archival analysis remain in central clouds. Wu et al. proposed a hybrid MCC-MEC offloading model and a distributed deep-learning-driven algorithm for heterogeneous clouds. Their reported experiments showed that the proposed Deep-learning-Driven Task-Offloading (DDTO) approach could improve computational performance by up to 63% relative to a local-only scheme. That value is a result of their stated experimental configuration and should not be generalized without replication.

2.5 | Research Gap

Much of the literature optimizes latency and energy but treats privacy, failure recovery, fairness, and model uncertainty as secondary concerns. A city-scale system also faces non-stationary demand, geographic imbalance, device churn, and administrative boundaries. A stronger formulation should therefore combine performance objectives with hard feasibility constraints, privacy-aware placement, robust capacity limits, interpretable monitoring, and reproducible evaluation [6].

3 | Collaborative Architecture and System Model

3.1 | Three-Tier Architecture

Table 2. Logical data and control path: IoT devices <-> edge/MEC servers <-> central cloud.

IoT Device Layer	Edge/MEC Layer	Central Cloud Layer
Sensing, local preprocessing, urgent fallback, lightweight inference	Nearby execution, aggregation, caching, stream analytics, radio-aware scheduling	Elastic computing, global analytics, model training, archival storage
Examples: cameras, meters, vehicles, gateways	Examples: roadside units, base-station servers, district micro-datacenters	Examples: municipal cloud, public cloud, regional datacenter

The architecture separates the data plane from the control plane. The data plane transports task inputs, intermediate features, and results. The control plane collects resource states, applies admission and placement policies, allocates bandwidth and CPU shares, and records audit events. Local execution remains available as a safe fallback when connectivity is disrupted [7].

3.2 | Task and Resource Definitions

Consider N IoT devices indexed by i and M edge servers indexed by m . Task i is represented by the tuple $\tau_i = (d_i, c_i, o_i, L_i, s_i)$, where d_i is the input size in bits, c_i is the required CPU cycles, o_i is the output size, L_i is the maximum acceptable end-to-end delay, and s_i is a sensitivity or placement label. The local CPU frequency is f_i^L , the allocated edge frequency is f_{im}^E , and the cloud allocation is f_i^C .

The binary variables x_i^L , x_{im}^E , and x_i^C indicate local, edge-server- m , and cloud execution. Exactly one destination is selected:

$$x_i^L + \sum_{m=1}^M x_{im}^E + x_i^C = 1, \quad x \in \{0,1\}. \quad (1)$$

3.3 | Communication Model

For uplink bandwidth b_{im} , channel gain h_{im} , transmit power p_i , and noise spectral density N_0 , an achievable wireless rate can be approximated by the Shannon expression:

$$r_{im} = b_{im} \log_2 \left(1 + \frac{p_i h_{im}}{N_0 b_{im}} \right). \quad (2)$$

The corresponding input-upload delay to edge server m is $t_{im}^{\text{up}} = \frac{d_i}{r_{im}}$. If the task is forwarded from the edge to the central cloud, let R_{mC} denote the backhaul rate and δ_{mC} the propagation and routing delay. The additional cloud-transfer delay is:

$$t_{im}^{\text{bh}} = \frac{d_i}{R_{mC}} + \delta_{mC}, \quad (3)$$

For response size o_i , the downlink time may be modeled similarly. In many inference applications o_i is much smaller than d_i , but it should not be omitted when tasks return maps, images, or control plans.

3.4 | Computation Delay

$$t_i^L = \frac{c_i}{f_i^L}, \quad (4)$$

$$t_{im}^E = \frac{d_i}{r_{im}} + \frac{c_i}{f_{im}^E} + \frac{o_i}{r_{mi}^{\text{down}}}, \quad (5)$$

$$t_i^C = \frac{d_i}{r_{im}} + t_{im}^{\text{bh}} + \frac{c_i}{f_i^C} + \frac{o_i}{R_{Cm}} + \frac{o_i}{r_{mi}^{\text{down}}}. \quad (6)$$

The realized end-to-end latency is the destination-weighted sum:

$$T_i = x_i^L t_i^L + \sum_m x_{im}^E t_{im}^E + x_i^C t_i^C. \quad (7)$$

3.5 | Energy and Cost Model

Dynamic voltage and frequency scaling commonly motivates a local-computation energy approximation proportional to the square of CPU frequency. With effective switched-capacitance coefficient κ_i :

$$E_i^L = \kappa_i c_i (f_i^L)^2. \quad (8)$$

Transmission energy is the product of power and airtime. For edge execution, $E_{im}^E = p_i d_i / r_{im}$. Cloud offloading uses similar device transmission energy but may incur additional monetary or network cost. A generalized cost can include compute price π , bandwidth price μ , and a privacy-risk penalty $\rho_i(a_i)$:

$$C_i = \pi_a c_i + \mu_a d_i + \rho_i(a_i), \quad a_i \in \{L, E, C\}. \quad (9)$$

4 | Optimization Problem

4.1 | Multi-Objective Utility

A city operator may minimize a weighted combination of normalized latency, device energy, monetary cost, and deadline violations. Normalization prevents one unit from dominating solely because of scale. Let T_{ref} , E_{ref} , and C_{ref} be reference values and let $[z]^+ = \max(z, 0)$. The objective is:

$$\min_{x,b,f} \sum_i \left[\alpha \frac{T_i}{T_{\text{ref}}} + \beta \frac{E_i}{E_{\text{ref}}} + \gamma \frac{C_i}{C_{\text{ref}}} + \eta \frac{[T_i - L_i]^+}{L_i} \right]. \quad (10)$$

The nonnegative weights α , β , γ , and η encode policy priorities. Safety-critical traffic control may assign a high latency-violation weight, while battery-powered environmental sensors may emphasize energy.

4.2 | Constraints

$$\sum_i b_{im} \leq B_m, \quad b_{im} \geq 0, \quad (11)$$

$$\sum_i f_{im}^E \leq F_m^E, \quad f_{im}^E \geq 0, \quad (12)$$

$$\sum_i f_i^C \leq F^C, \quad (13)$$

$$T_i \leq L_i \quad \text{for hard-deadline tasks}, \quad (14)$$

$$x_i^C = 0 \quad \text{if sensitivity policy } s_i \text{ forbids cloud transfer}. \quad (15)$$

Constraints (11)-(13) enforce bandwidth and compute capacities. Eqs. (14) and (15) represent service-level and governance requirements. Reliability can be added through chance constraints such as $\Pr(T_i \leq L_i) \geq 1 - \epsilon_i$, where ϵ_i is the permitted violation probability.

4.3 | Computational Nature

The joint problem contains binary placement variables, continuous bandwidth and CPU variables, nonlinear rate expressions, and coupled capacity constraints. It is therefore generally non-convex and combinatorial. Exhaustive enumeration grows exponentially with the number of devices. This motivates decomposition, approximation, and learning-assisted decision rules, while a feasibility layer remains necessary to guarantee capacity and placement constraints [8], [9].

5 | Distributed Deep Learning-Driven Task Offloading

5.1 | State, Action, and Policy

At decision epoch t , device i observes a local state vector $z_i(t)$ containing task size, cycle demand, deadline, battery state, channel quality, recent edge load, and a privacy label. The action $a_i(t)$ selects local, one of the reachable edges, or central cloud. A neural policy π_θ maps the state to action scores. Distributed execution reduces the need to send raw observations to a central scheduler.

$$a_i(t) = \operatorname{argmax}_a \pi_\theta(a | z_i(t), g(t)). \quad (16)$$

Here $g(t)$ is a compact coordination signal such as advertised server price, congestion level, or remaining capacity. The signal enables devices to react to shared bottlenecks without revealing full local data.

5.2 | Training Objective

Training labels may be generated by solving smaller offline optimization instances, by using a trusted heuristic, or through reinforcement learning. For supervised imitation, a cross-entropy term can be combined with penalties for infeasible or deadline-violating actions:

$$L(\theta) = - \sum_i \sum_a y_{ia} \log \pi_\theta(a | z_i) + \lambda_1 P_{\text{cap}} + \lambda_2 P_{\text{deadline}} + \lambda_3 \|\theta\|_2^2. \quad (17)$$

For reinforcement learning, the reward can be the negative weighted system cost in Equation (10), with strong penalties for hard-constraint violations. Training should randomize workload, channel, and failure conditions to reduce overfitting to one city district or traffic pattern.

5.3 | Distributed Coordination and Feasibility Repair

Independent device predictions can collectively exceed an edge server capacity. A feasibility repair stage ranks requests by marginal utility, deadline urgency, and privacy eligibility. Accepted tasks receive bandwidth and CPU allocations; rejected tasks are redirected to another feasible edge, the central cloud, or local fallback. This separation allows fast learned inference while preserving operational constraints [10].

A practical decision cycle is: 1) devices summarize their local states; 2) edge servers broadcast capacity indicators; 3) local DDTO agents produce candidate destinations; 4) a lightweight coordinator resolves conflicts; 5) resources are allocated; and 6) outcomes are logged for monitoring and periodic retraining.

5.4 | Complexity Discussion

If a neural network has layer widths $n_0, \dots, n_K$, one forward pass requires approximately $O(\sum_{k=1}^K n_{k-1} n_k)$ multiply-accumulate operations. For fixed architecture, inference scales approximately linearly with the number of active devices. In contrast, exhaustive destination search considers up to $(M+2)^N$ assignments before continuous resource allocation. The learned policy therefore offers a significant complexity advantage, although training and feasibility repair add overhead.

6 | Challenges and Limitations

Table 3 summarizes the key challenges and limitations of intelligent edge and IoT systems, their scientific implications, and potential mitigation strategies. These considerations provide a basis for improving system performance, reliability, security, and practical implementation.

Table 3. Expanded limitations and mitigation directions.

Challenge	Scientific implication	Mitigation direction
Privacy and security	Raw sensor streams may identify people, locations, or critical infrastructure.	Encryption, identity management, least privilege, local feature extraction, audit logs, retention limits.
Interoperability	Incompatible protocols, data models, and APIs impede cross-vendor orchestration.	MQTT/CoAP gateways, standardized APIs, semantic models, conformance testing.
Resource variability	Wireless rate and edge load change quickly.	Online monitoring, robust optimization, adaptive policies, admission control.
Scalability	Large device populations create signaling and scheduling overhead.	Hierarchical control, clustering, distributed inference, event-driven reporting.
Reliability	Edge failure or network partition can interrupt critical services.	Replication, failover routes, checkpointing, local safe mode, disaster recovery.
Fairness	Utility optimization can repeatedly disadvantage weak-channel devices or low-income districts.	Minimum service guarantees, fairness terms, district-level monitoring.
Model drift	Learned policies degrade when workload or network distributions change.	Drift detection, periodic validation, controlled retraining, rollback.
Public trust	Opaque collection and automated decisions can reduce acceptance.	Transparency, purpose limitation, citizen consultation, appeal mechanisms.
Implementation cost	Edge deployment and maintenance require capital and skilled staff.	Phased rollout, shared infrastructure, total-cost-of-ownership analysis.

7 | Implementation and Security Solutions

7.1 | Privacy-by-Design and Cryptographic Protection

Data should be classified before placement decisions are made. Encryption in transit and at rest, device authentication, key rotation, signed software updates, and tamper-evident logs form a baseline. Privacy-sensitive streams can be transformed at the edge into counts, alerts, or embeddings so that the central cloud receives only the minimum necessary information. The offloading optimizer should enforce these placement restrictions as hard constraints rather than optional penalties.

7.2 | Middleware and Interoperability

A middleware layer can normalize device protocols, schemas, timestamps, units, and identities. MQTT is suitable for publish-subscribe telemetry, while CoAP supports constrained devices and request-response interactions. An interoperable design also requires versioned APIs, schema registries, capability discovery, and testing across vendors. Protocol translation alone is insufficient if two systems attach different meanings to the same field.

7.3 | Dynamic Resource Allocation

Microservices and containerized execution can isolate application components and scale them independently. Telemetry should measure queue lengths, CPU/GPU utilization, memory pressure, wireless quality, and deadline misses. Autoscaling must account for startup delay, otherwise capacity may arrive after a short-lived demand spike. Priority admission is appropriate for emergency workloads, but starvation controls are necessary for routine public services.

7.4 | Resilience and Continuity

Critical functions should not depend on a single edge node or network path. Replication across nearby edge sites, health checks, graceful degradation, cached models, and local safe-mode logic support continuity. Recovery objectives should be defined per application because a traffic signal controller and an archival dashboard have different tolerances for interruption and data loss.

7.5 | Workforce and Governance

Technical deployment should be paired with training in distributed systems, cybersecurity, data engineering, machine learning operations, and public-sector governance. Responsibility for model approval, incident response, data retention, vendor access, and citizen complaints should be documented. Governance metrics belong beside latency and energy metrics in operational dashboards.

8 | Evaluation Methodology

8.1 | Baselines and Scenarios

A reproducible evaluation should compare at least local-only, edge-only, cloud-only, greedy minimum-latency, optimization-based, and DDTO policies. Scenarios should vary device count, task input size, CPU-cycle demand, edge capacity, bandwidth, channel quality, backhaul delay, privacy restrictions, and failure events. Random seeds and workload distributions should be reported.

8.2 | Performance Metrics

$$\text{Average latency} = \frac{1}{N} \sum_i T_i, \quad (18)$$

$$\text{Deadline satisfaction ratio} = \frac{1}{N} \sum_i 1 [T_i \leq L_i], \quad (19)$$

$$\text{Energy per completed task} = \frac{\sum_i E_i}{\sum_i 1 [\text{task } i \text{ completed}]}, \quad (20)$$

$$\text{Jain fairness index} = \frac{(\sum_i q_i)^2}{N \sum_i q_i^2}. \quad (21)$$

Additional metrics include the 95th- and 99th-percentile latency, bandwidth consumption, server utilization, monetary cost, decision time, infeasible-action rate, recovery time after failure, privacy-policy violations, and model accuracy under distribution shift. Jain's index can be applied to per-device service rates q_i to expose unequal treatment hidden by good average performance.

8.3 | Experimental Integrity

Results should include confidence intervals or repeated-run variability rather than single averages. Hyperparameters, hardware, simulator settings, code versions, and stopping criteria should be documented. Ablation studies should remove coordination signals, feasibility repair, privacy constraints, or selected state features to identify which components generate performance gains. A claim such as “63% improvement” must define the metric, baseline, workload, and calculation.

9 | Illustrative Analytical Example

Consider one task with $d = 8$ Mbit, $c = 1.2 \times 10^9$ cycles, a 20 Mbit/s uplink, local CPU $f^L = 0.6$ GHz, edge allocation $f^E = 4$ GHz, cloud allocation $f^C = 12$ GHz, and 80 ms edge-to-cloud transfer overhead. Ignoring a small return payload, local latency is 2.0 s. Edge latency is $8/20 + 1.2/4 = 0.70$ s when units are consistently converted. Cloud latency is $8/20 + 0.08 + 1.2/12 = 0.58$ s. In this simplified snapshot, cloud execution has lower latency than edge execution because its compute advantage exceeds the added backhaul delay.

The conclusion changes when backhaul delay rises, when cloud price is high, when sensitive data cannot leave the district, or when the edge has a GPU that accelerates the task. This example demonstrates why a static “always edge” or “always cloud” rule is inadequate. It is illustrative only and is not presented as experimental evidence.

10 | Discussion and Future Research

Collaborative offloading should be treated as a socio-technical control problem rather than a narrowly computational one. Latency and energy are measurable, but legitimate deployment also requires security, transparency, resilience, accessibility, and equitable service. Future research can explore federated or split learning to reduce raw-data movement, graph neural networks to represent device-edge topology, digital twins for safe policy testing, robust optimization under uncertain demand, and carbon-aware placement that shifts delay-tolerant workloads toward lower-emission resources.

Another priority is transferability. A policy trained for one district may encounter different radio conditions, server hardware, mobility patterns, or administrative rules elsewhere. Meta-learning, domain adaptation, uncertainty estimation, and conservative fallback policies can reduce risk. Human operators should retain visibility into constraint violations, unusual actions, and model drift.

11 | Conclusion

This paper expanded the original smart-city IoT discussion into a structured collaborative edge-cloud framework. A three-tier architecture allows tasks to execute locally, at nearby edge servers, or in central clouds. The mathematical formulation integrates communication and computation delay, device energy, monetary cost, capacity, deadlines, and privacy placement. A DDTO-oriented policy can approximate high-quality destination choices at low inference cost, while a coordination and feasibility layer prevents aggregate capacity

violations. The analysis also shows that technical performance alone is insufficient. Interoperability, security, resilience, fairness, governance, and reproducible evaluation are essential to scientific and operational credibility. The reported 63% gain is retained only as an attributed finding from Wu et al., not as a result newly produced by this manuscript. Future empirical work should validate the proposed formulation across realistic workloads, network traces, failures, and city governance constraints.

Author Contributions

S. H.: Conceptualization, methodology, original draft preparation, and identification of limitations and solution directions. A. M.: revised mathematical modeling, software, validation, visualization, or supervision.

Funding

This research received no external funding.

Data Availability

No new dataset was generated or analyzed in this expanded conceptual manuscript. Any future experimental study should provide the dataset or simulation configuration, subject to privacy and licensing constraints.

Conflicts of Interest

The author declares no conflict of interest.

References

- [1] Kong, L., Tan, J., Huang, J., Chen, G., Wang, S., Jin, X., ... Das, S. K. (2022). Edge-computing-driven internet of things: A survey. *ACM Computing Surveys*, 55(8), 1–41. <https://doi.org/10.1145/3555308>
- [2] Cao, B., Li, Z., Liu, X., Lv, Z., & He, H. (2023). Mobility-aware multiobjective task offloading for vehicular edge computing in digital twin environment. *IEEE Journal On Selected Areas in Communications*, 41(10), 3046–3055. <https://doi.org/10.1109/JSAC.2023.3310100>
- [3] Cao, K., Hu, S., Shi, Y., Colombo, A. W., Karnouskos, S., & Li, X. (2021). A survey on edge and edge-cloud computing assisted cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 17(11), 7806–7819. <https://doi.org/10.1109/TII.2021.3073066>
- [4] Meruje Ferreira, L. M., Coelho, F., & Pereira, J. (2024). Databases in edge and fog environments: A survey. *ACM Computing Surveys*, 56(11), 1–40. <https://doi.org/10.1145/3666001>
- [5] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5), 637–646. <https://doi.org/10.1109/JIOT.2016.2579198>
- [6] Kanupriya, Chana, I., & Goyal, R. K. (2024). Computation offloading techniques in edge computing: A systematic review based on energy, QoS and authentication. *Concurrency and Computation: Practice and Experience*, 36(13), e8050. <https://doi.org/10.1002/cpe.8050>
- [7] Bozorgchenani, A., Tarchi, D., & Corazza, G. E. (2018). A control and data plane split approach for partial offloading in mobile fog networks. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1–6). IEEE. <https://doi.org/10.1109/WCNC.2018.8377170>
- [8] Lin, Z., Bi, S., & Zhang, Y. J. A. (2020, December). Optimizing AI service placement and computation offloading in mobile edge intelligence systems. In *GLOBECOM 2020-2020 IEEE Global Communications Conference* (pp. 1–7). IEEE. <https://doi.org/10.1109/GLOBECOM42002.2020.9348232>
- [9] Acheampong, A., Zhang, Y., Xu, X., & Kumah, D. A. (2023). A review of the current task offloading algorithms, strategies and approach in edge computing systems. *Comput. Model. Eng. Sci*, 134(1), 35–88. <https://doi.org/10.32604/cmes.2022.021394>
- [10] Sahni, Y., Cao, J., Yang, L., & Wang, S. (2022). Distributed resource scheduling in edge computing: Problems, solutions, and opportunities. *Computer Networks*, 219, 109430. <https://doi.org/10.1016/j.comnet.2022.109430>